AFIT/GOR/ENS/99M - 05

A COMPARISON OF GENETIC ALGORITHM PARAMETRIZATION ON SYNTHETIC OPTIMIZATION PROBLEMS

THESIS

Mehmet Eravsar, Lieutenant,TUAF

AFIT/ENS/99M - 05

Approved for public release; distribution unlimited

AFIT/GOR/ENS/99M - 05

Disclaimer

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense, the US Government or Turkish Air Force.

AFIT/GOR/ENS/99M - 05

A COMPARISON OF GENETIC ALGORITHM PARAMETRIZATION ON SYNTHETIC OPTIMIZATION PROBLEMS

THESIS

Presented to the Faculty of the Graduate School of Engineering

Of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Mehmet Eravsar

Lieutenant, TUAF

March 1998

AFIT/GOR/ENS/99M - 05

A COMPARISON OF GENETIC ALGORITHM PARAMETRIZATION ON SYNTHETIC OPTIMIZATION PROBLEMS

Mehmet Eravsar

Lieutenant, TUAF

Approved:

Raymond R. Hill Jr. — 12 Mar 99

Major Raymond R. Hill Jr. — Date

Advisor

James T. Moore — 12 Mar 99

James T. Moore, Ph.D — Date

Reader

Acknowledgements

First of all, I would like to thank the Turkish People whose taxes paid for my top of the line OR education and let me meet high caliber officers and friends at AFIT.

A Turkish proverb says, "Volunteer to be a slave for forty years for someone who teaches even a single letter." Without exaggeration, this proverb is not even close to describe my feelings and wishes for the faculty who worked hard to teach and stood as a vivid example of professionalism. Among the faculty special thanks to my advisors, Maj. Hill and Dr. Moore, who not only helped me with their expertise in OR but with their never ending thoughtfulness and friendliness. Maj Hill, despite his "already full white-board," was very generous and thoughtful to offer a worthwhile thesis topic. His assistance was vital to my graduating.

Finally, among all the wonderful people I met, I want to thank the one who really sacrificed the most, my wife Esra. We got married the day I left Turkey to come to AFIT. Since then she allowed me to spend the most valuable days of our lives at AFIT while she was alone and separated from me. There is nothing I can do to make up for the lost time, but I promise I will try.

My next project is to construct a real-world model which will remember all these wonderful people and their efforts to work for peace, humanity and friendship. This will keep me busy to the last day of my life.

Table of Contents

List of Figures

List of Tables

Abstract

Meta-heuristics have been deployed to solve many hard combinatorial and optimization problems. Parameterization of meta-heuristics is an important challenging aspect of meta-heuristic use since many of the features of these algorithms can not be explained theoretically. Experiences with Genetic Algorithms (GA) applied to Multidimensional Knapsack Problems (MKP) have shown that this class of algorithm is very sensitive to parameterization. Many studies use standard test problems, which provide a firm basis for study comparisons but ignore the effect of problem correlation structure.

This thesis applies GA to MKP. A new random repair operator, which projects infeasible solutions into feasiblity, is proposed. This GA application is tested with synthetic test problems, which map possible correlation structures as well as possible slackness settings. Effect of correlation structure on solution quality found both statistically and practically significant. Depending on the Response Surface Methodology design, proposed is a GA parameter setting which is robust in both solution quality and computation time.

A COMPARISON OF GENETIC ALGORITHMS' PARAMETERIZATION ON SYNTHETIC OPTIMIZATION PROBLEMS

Chapter 1. Motivation and Outlines

Meta-heuristics have been used to solve many hard combinatorial and optimization problems. Parameterizations of meta-heuristics are an important and challenging aspect of meta-heuristic use since many of the features of these algorithms can not be explained theoretically. Deficiencies in analytic approaches mean empirical studies are conducted to examine parameterization of meta-heuristics. This emerging "empirical science" is addressed in Hooker (1994).

Experience with Genetic Algorithms (GA) has shown that this class of algorithm is very sensitive to parameterization. This has been true in GA studies applied to Multidimensional Knapsack Problems (MKP), for instance Theil and Voss (1994). Many studies use standard test problems which provide a firm basis for study comparisons but ignore the effect of problem correlation structure. Freville and Plateau (1996) and Hill (1997) infer heuristic and enumerative algorithm performance when correlation exists in test problems. This research extends Hill's results to a GA empirical study.

We develop a GA which inherits its operators from GALib©. We exploit results from previous studies, but define an operator to repair infeasible solutions both within the initial population and in subsequent populations. Hill's (1997) test problems are used in the empirical study.

The main body of this thesis follows the format of a stand-alone article. This is provided in Chapter Two. Chapter Three outlines future avenues that we find necessary to complete and improve upon this study. Details, which are excluded from the article, are found in Appendices. Appendix A includes a detailed review of past studies. Detailed analysis of data that is generated in our study can be found in Appendix B. Finally, Appendix C includes source code for the GA we implemented.

Chapter.2. ARTICLE

1. Introduction.

1.1. Purpose of This Study

Among the heuristic solution techniques for 0-1 Multi-Dimensional Knapsack Problems, Genetic Algorithms (GA) are important due to their speed and high quality solutions. As in many other heuristic techniques, selection of algorithm parameters plays a significant factor in GA performance. However, there is not a general theorem that explains why GA's have the features that they have (Beasly, *et al.* 1993). This lack of proven theoretical knowledge encourages empirical studies addressing GA parameterization.

This paper is an empirical study of GA's applied to the Two-dimensional Knapsack Problem (2KP). Hill and Reilly's (1997) test problems are used for the study. These problems vary problem correlation structure and right hand side ratios (i.e., constraint tightness). We seek robust GA parameter settings for these problems. Furthermore, these GA parameters are used to examine the effect of problem correlation structure and slackness.

1.2. Multi-Dimensional Knapsack Problems

In a knapsack problem, we seek the subset of items which maximize the profit while not exceeding the resource constraint. There are a variety of knapsack problems. In 0-1 Knapsack Problems (0-1 KP) each item is selected once. In Bounded Knapsack Problems (BDP) each item is selected a limited number of times. In Multiple Knapsack

Problems more than one knapsack is filled simultaneously (Bjorndal, 1995). Details on knapsack problems are found in Martello and Toth (1990).

Another type of knapsack problem is called 0-1 Multidimensional Knapsack Problem (MKP). In MKP, variables take values of 0 or 1 and must satisfy more than one constraint. This type of problem is formulated as follows:

$$\max \sum_{j=1}^{n} c_j \cdot x_j \tag{1}$$

subject to

$$\sum_{j=1}^{n} a_{i,j} \cdot x_j \leq b_i \qquad i = \{1,2..m\} \tag{2}$$

$$x_j = \{0,1\}$$

where

$c_j > 0, a_{i,j} \geq 0$, and at least one $a_{i,j} > 0$ for each j.

Cutting stock, resource allocation and capital budgeting problems are among the real world applications of 0-1 MKP. Freville and Plateau (1996) provide a comprehensive overview of MKP applications.

MKP is a NP-hard problem (Frieze and Clark, 1984), so heuristics are a favorable means of solving MKP instances.

1.3. Genetic Algorithms

In 1975 Holland introduced the basis of Genetic Algorithms (GA). The principles he presented were sufficient to create optimization algorithms based on a genetics analogy.

In nature, "survival of the fittest" means highly adapted species are selected to produce subsequent generations. It is expected that the average fitness or adaptation of each generation improves over that of the current generation, because each generation inherits good features from the current generation. Conceptually, this means we would expect to eventually achieve a population composed of identical individuals. However, mutation occurs and introduces diversification in the population and prevents this homogenous situation.

Analogously, GA solves optimization problems by simulating nature's generation paradigm. A typical GA starts with an initial population consisting of random solutions or individual chromosomes. The fitness of each chromosome is evaluated by an appropriate measure, such as an objective function value for an optimization problem. These chromosomes mate based on their fitness in the population. This mating produces an offspring (new solution) inheriting features from each parent's chromosome. The bits within this chromosome mutate based on a small probability to ensure diversity. These offspring replace older chromosomes in the population that are not as fit as the offspring. In a given generation, this mating and replacement process is carried out until the number of replacements desired in the population is achieved. The number of generations simulated determines when the heuristic terminates.

Each generation produced during the evolution of a GA is expected to contain fitter chromosomes or solutions than past generations. For optimization applications, these are improved objective function values. This is aided by selecting the fittest chromosomes to mate. This selection criterion encourages retaining the good features of

chromosomes in new chromosomes. Beasley *et al* (1993) claim that a properly designed GA will converge to the global optimality.

GA terminology is consistent with natural genetics. The *Chromosome* is the main object. Each chromosome represents a possible solution to the given problem. *Genes* are the small units comprising chromosomes and may be the bit stream of the solution. Location of each gene is called *loci* and genes may have a set of values, which are called *alleles*. Mating is accomplished using crossover; in one point crossover an offspring chromosome is derived with the first part from parent 1 and the second part from parent 2. The parts are determined by the crossover point.

A typical GA follows the following procedure:

```
Initialize;
     Generate initial population
     Calculate the fitness and relative fitness of
     initial chromosomes

Evolve while termination criterion is not satisfied;
     Reproduce until population is replaced by desired
     amount
          Select parents according to their relative
          fitness.
          Crossover parents at random points with
          desired probability
          Mutate at given probability
     Evaluate fitness and calculate relative fitness of
     offspring
     Replace offspring with the selected ancestors
End.
```

Chromosome formatting is decided based on the tpe of coding. Binary coding using 0 or 1 value bits is analogous to natural genetics and is commonly used but has some drawbacks if the chromosome length is long. Gray and integer coding are also used. Gray coding is invented to lessen the distance gaps between phenotype and genotype of a chromosome. For example, in a 4-bit chromosome, phenotypes 16 and 15 are adjacent to

each other but in genotype space they are away from each other (1111 and 1110). In gray coding genotype for 16 and 15 can be mapped by one byte difference (1000 and 1001, respectively). Although gray coding helps to reduce distance between genotypes, there is no simple algorithm for decoding gray coding into phenotypes (Reeves, 1993). Furthermore, they are not critical in combinatorial optimization applications of GA.

Integer coding is used when the binary coding operators are not applicable to the solution of problem (for instance, Traveling Salesman Problem). Each gene can take integer values varying from 1 to the chromosome length. Different strategies can be followed to transfer a chromosome into a solution. Integer coding introduces new crossover and mutation operators that are different from binary operators.

Population initialization is either random or seeded. Random populations consists of randomly generated chromosomes. Seeding uses auxiliary algorithms to generate good feasible solutions which are included in the initial population. Both techniques have advantages and disadvantages. In random initialization, the entire solution space is considered. This diversifies the population causing a longer time to converge to an optimal solution but this may avoid local optima traps. On the other hand, seeding an initial populations with some well known chromosomes or solutions can accelerate convergence but may also cause premature convergence, returning local versus global optima (Reeves, 1993; Davis, 1991).

The fitness function assigns a value to each chromosome to represent the chromosome's goodness or quality. In unconstrained function optimization, this is the function value. Unfortunately, in constrained optimization problems this requires

strategies to incorporate feasibility into the fitness value and rules guiding if infeasible solutions are allowed in a population.

Selection for mating is performed in many different ways. A common characteristic of these different techniques is that the number of times a chromosome is selected is proportional to its relative fitness value. A simple selection allocates probability to each chromosome by using a roulette wheel with slot sizes proportional to the relative fitness value of the chromosome. This is the probability that the chromosome is selected as a parent. Selection occurs as many times as required to select enough parents to generate the next generation. Tournament selection randomly indexes chromosomes with numbers 1 through population size (P). Two randomly chosen parents are compared and the best one is selected as the candidate for mating. Again selection continues until the required numbers of parents are selected.

Crossover is an operator that guarantees different offspring from their parents. One-point, two-point and uniform crossovers are illustrated in (Figures 1-3). In one-point crossover, a random loci is selected and genes located to the right hand of this loci are interchanged with the genes of the other parent. The result is two new different chromosomes which inherit some genes from each parent. In two-point crossover two locis are randomly selected and the part of the chromosome located between these locis are interchanged. Crossovers based on more than two points can be devised extending this same procedure.

Uniform crossover is a multipoint crossover strategy. A string of binary genes is randomly produced the same length as the chromosome length and if the binary number is 0, the corresponding gene value for the child is inherited from the first parent;

otherwise, it comes from the second parent To produce a second child, reverse the roles of 0 and 1 in the gene selection process. Crossover details for the MKP are in Section 4.

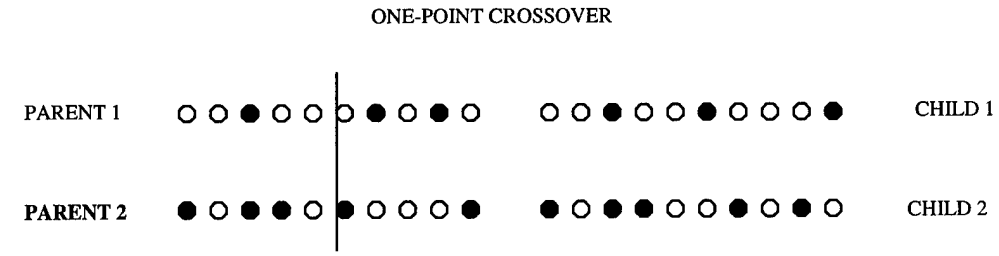


Figure 1. Onepoint Crossover

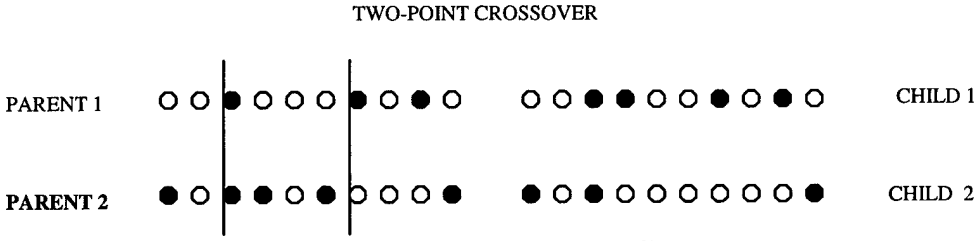


Figure 2. Two-point Crossover

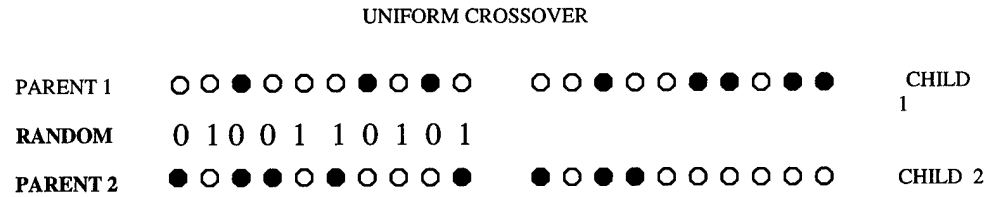


Figure 3. Uniform Crossover

The mutation operator serves to diversify the population by changing genes of a new chromosome according to a probability of mutation. Mutation helps explore unvisited regions of the solution space (i.e, diversification).

GA effectiveness has been best explained with the building block hypothesis and the schema theorem both of which are related to schema. A schema is a string of values {0,1,*}. Schema explores the similarities between chromosomes. A schema represents all

strings, which match it on all positions other than the genes with allele (*) (Michalewicz, 1992). The order of a schema is defined by the number of ones in its structure. The defining length is the distance between the outermost * elements or genes.

The schema theorem says that short low-order, above–average schemata will produce with probability that increases exponentially with the generation number. This guarantees that chromosomes with well schema produce more often. Building blocks are short defined length of schemata that work good when combined together into a chromosome. This hypothesis requires that related genes must be close together in a chromosome and interaction between genes must be very low. However, finding chromosomes bearing these two properties is not always possible.

2. Combinatorial Optimization and Heuristics.

2.1. Combinatorial Problems and Optimization

Combinatorial problems consist of arrangements of objects (scarce resources) to meet some desired objectives. Integrality is a characteristic of these objects. Thus, the number of possible arrangements is finite. The travelling salesman problem (TSP), parallel machine scheduling, the knapsack problem (KP), portfolio selection, capital budgeting, facility location, design and production of VLSI circuits, political districting, set covering and assignment problems are classes of combinatorial optimization problem (Karla, 1996; Nemhauser, 1988).

Parker (1988) defines four categories of combinatorial problems depending on the approach in answering the questions:

1. Existence, are there specific arrangements?
2. Evaluation, how good are arrangements?
3. Enumeration, how many arrangements exist?
4. Extremization, is there a best arrangement?

Combinatorial optimization (CO) seeks an optimal arrangement among the finite alternatives. When the problem size is small, even simple enumeration tools can find optimal solutions in acceptable time. However, the computational effort required to identify the optimal arrangement from among these finite alternatives grows exponentially with the number of variables. For example, a problem of 100 binary variables has 1.2×10^{30} possible solutions. This number doubles with each additional variable introduced into the problem. For even moderate size problems, enumeration techniques are not computationally efficient.

Problem solving techniques for these problems can be classified into the following four categories:

1. Enumeration. As stated earlier, small instances of problems can be solved by enumerating all possible solutions. However, it is computationally intensive for even moderate size problems. Branch and Bound algorithms are the most common enumeration type of technique.
2. Relaxation and Decomposition Techniques. Besides the LP relaxation of an integer problem, another relaxation of the integer programming problem can be achieved by integrating some or all of constraints into the objective function. This type of relaxation, Lagrangian Relaxation, provides tighter bounds than the LP solution to the same problem (Hoffman, 1996). Decomposition techniques divide a

main problem into easy separate problem subsets and combine the solution of subsets to obtain a solution for the main problem.

3. Cutting Plane Algorithms. These algorithms are devised to solve an IP using an LP relaxation but suffer from slow convergence. The algorithm can be summarized as follows. First, find an LP optimum of the relaxed IP. Second, pick a constraint which cuts the LP optimum out of the solution space but does not eliminate feasible integer solutions. Add this cutting constraint to the problem a cutting constraint. Solve the new problem by dual simplex. If integer solutions are found any time in the algorithm flow, the algorithm stops with the optimal solution.

4. Heuristic Techniques. While the above techniques seek an optimal solution, heuristic techniques merely seek good solutions. Increased computing speed and improved heuristic algorithms have made these techniques increasingly popular in research and application.

2.2. Heuristics

The word "heuristics" is derived from the Greek word "heuriskein" meaning "to discover." In this sense, heuristics (also called approximation algorithm or inexact solution) are described in Barr, *et al* (1995) as:

> " a well defined set of steps for quickly identifying a high-quality solution for a given problem where a solution is a set of values for the problem unknowns and quality is defined by a stated evaluation metric or criterion ."

Another definition of heuristics is Reeves' (1993) :

> "... a technique which seeks good (i.e. near optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is."

Fisher, *et al.* (1983) sees the 1950s as the flourishing and attractive years of heuristics, the 1960s as the return of exact optimization algorithms and the 1970s as the disappointment years of the exact algorithm studies since computational complexity studies proved that many of the algorithms devised were inefficient when used to solve hard problems. Since the 1980s, intellectual energy has been invested primarily on the heuristics studies.

Many tutorials try to explain the reasons for implementing heuristics as they become more popular tools in optimization. Two of the most detailed ones are Zanakis and Evans (1981) and Silver, *et al.* (1980). Their findings can be summarized as follows:

- Problems may have neither an analytical nor an iterative solution procedure.
- Although an exact algorithm exists, one may not be able to afford time and storage requirements.
- Inexact or limited data used to estimate model parameters might inherently contain errors much larger than the near optimality (sub-optimality) of a good heuristic.
- An exact algorithm is not available due to its cost or hardware limitations.
- As a starting point for the other exact or inexact algorithms. Sometimes heuristics are used to alleviate the burdens on another algorithm even during the operation of that algorithm.
- They are simple and understandable when compared with the other algorithms.

These reasons do not preclude the use of exact algorithms. If an exact algorithm is available, resources can be allocated for its use. However, in large-scale problems, small deviations from the optimal solution may be practically insignificant. In such cases, other merits of both exact algorithms and heuristics must be compared to make a decision about which one to use.

Many heuristics are documented in the literature. Although, most of them are problem specific, they can be classified according to the philosophy used in their structure. Silver *et al.* (1980) classifies the heuristics as:

- Decomposition methods. The main problem is broken into small parts and each small part is solved separately.
- Inductive Methods. In these methods, solution of a smaller or simpler problem is generalized for bigger or harder ones. Properties of the solution for simpler cases may be used to develop a heuristic for the more general case.
- Feature Extraction Methods. First, the optimal solutions to several numerical cases under consideration are obtained and then the common characteristics of these solutions are extracted (reduced) and assumed to hold in general.
- Methods Involving Model Manipulation. The nature of the problem is perturbed in some way to expedite the solution and then the solution of the revised problem is used as a representative of the solution of the real problem. Good examples of manipulation include modification of the objective function, relaxation of certain constraints, and aggregation of variables.
- Constructive Methods. The main idea of these methods is to construct a single feasible solution, often in a deterministic sequential fashion. Greedy algorithms are of this class. These algorithms suffer from their myopic viewpoint of only considering the very next point.
- Local Improvement Methods. These methods are the most used ones among the listed method structures. In contrast with constructive methods, this method starts with a feasible solution and iteratively improves it. Meta-heuristics tend to be local improvement methods.

Modern heuristics or meta-heuristics have evolved in parallel with the improvements in computer hardware technology since 1970s. These relatively new methods are efficient approximation solution techniques for the problems that are found difficult or inefficient to solve with the earlier heuristics. Osman and Kelly (1996) defines meta-heuristics as

" an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficient near-optimal solutions"

Meta-heuristics are derived from many different areas such as classical heuristics, biology, metallurgy, artificial intelligence and neural networks. Nowadays, they are used extensively for cracking the hard problems efficiently at less computational cost. Some of the meta-heuristics that are developed and used in many areas are: genetic algorithms, simulated annealing, tabu search, GRASP (greedy random adaptive search procedure), problem-space search, neural networks, threshold algorithms and hybrids of these heuristics.

A limitation of meta-heuristics is their parameterization. When a heuristic is selected to solve a specific problem, poor heuristic parameterization can lead to local optima or poor solutions. Pilot runs may be needed to find a good parameter set for these meta-heuristics, but such runs may be costly and require too much time to conduct.

Most of the interesting papers on heuristics address parameterization. Chu and Beasely (1998), Theil and Voss (1993), Hoff, *et al.* (1998), Syswerda (1994), Hanafi and Freville (1998) and Schaffer, *et al.* (1994) are a few examples.

3. Past 0-1 MKP Heuristics

3.1. Greedy Heuristics

0-1 MKP greedy heuristics can be classified into two main groups as dual heuristics or primal heuristics. This classification is based on how they start searching for the solution, either infeasible or feasible, respectively. For MKP, a dual heuristic starts by selecting all items providing an infeasible solution while primal heuristics build a solution from no items selected. Senju and Toyoda (1968) and Magazine and Oguz

(1984) studied dual methods. Senju and Toyoda calculated an effective gradient for each element to evaluate each element's utility and deselected items until feasibility is obtained. Magazine and Oguz combine the Senju and Toyoda heuristic with Everett's Lagrange Multipliers to improve heuristic performance.

Primal Effective Gradient Method (PEGM) defined in Toyoda (1975) is a primal greedy algorithm which finds good approximate solutions to 0-1 programming problems. The method does not use enumeration at all. Instead, each variable's preferability was measured by an effective gradient. Variables are added to the solution according to their sorted effective gradient values until no further variables can be added.

To account for unbalanced resource usage in the constraints, Toyoda introduces origin moving. Origin moving increases resource usage penalties in the gradient function according to each constraint's resource availability. Origin moving was found efficient when the amount of resources used (constraint coefficients) were unbalanced.

Lee and Guignard (1988), Kochenberger, *et al.* (1974) and Loulou and Michalides (1979) modify the gradient calculations of Toyoda (1975) in an effort to improve heuristic performance.

3.2. Meta-Heuristics.

3.2.1. GA and MKP

Chu and Beasley (1998) proposed an algorithm that incorporates problem specific knowledge into a GA for solving 0-1 MKP. They generated synthetic test problems to test their algorithm.

Solutions are represented with binary genes. Tournament selection was used to pick parents and uniform crossover and flip mutation were used to generate new

chromosomes. They report that GA performance is insensitive to these operators and recommend random selection of these operators. Their population size was set to 100, with single replacement each generation and two bit mutation was used.

The objective function value was used to calculate fitness value and they used a heuristic to fix infeasible chromosomes. Their ADD/DROP heuristic operator fixes infeasible chromosomes, using a pseudo-utility ratio for knapsack problems calculated as the ratio of objective function coefficients (a_j) to the coefficients of the single knapsack constraint (c_j). Borrowing the surrogate duality approach of Pirkul (1987), they combine constraints to form a single constraint. Dual variables of the LP relaxation of the original problem are used as the surrogate multipliers.

Their ADD/DROP heuristic resembles the Senju and Toyoda (1968) approach. Items are dropped to make solutions feasible, and then some items are added back to improve the feasible solution if possible.

Chu and Beasly solved two different sets of problems to evaluate their algorithm performance. The first set, consisting of 55 problems, was taken from literature. These problems vary in number of variables (6-105), and number of constraints (2-30). Hill (1998) examines the correlation structure of these problems, and found the ranges of feasible correlation structures limited.

The second problem set was generated randomly and is available via the internet. They adopted the generation procedure of Freville and Plateau (1990). Thirty problems of each combination of constraint (5,10 and 30) and variables (100, 250 and 500) were generated, yielding 270 total test problems.

Optimal values for most of these problems are not known. Thus, the authors computed the solution quality by using the LP relaxation of each problem. Chu and Beasley found that GA is very effective in solving these MKP problems. However, they note that the larger the chromosome length and the smaller the slackness ratios, the less effective the GA. GA was stopped when 10^6 chromosomes were generated. They compare results with heuristics by Magazine and Oguz (1984) and Volgenant and Zoon (1990) and report that their GA dominates each.

Theil and Voss (1993) studied four different GA techniques. The first technique penalized the fitness function according to the level of feasibility and found that the degree to which the fitness of infeasible chromosomes are penalized is important. Too restrictive a penalty function and GA converges to sub-optimal values while too loose a penalty function allows infeasible solutions to dominate. Their penalty function had three parts, which evaluated the chromosome according to both their relative fitness value and their distance from feasibility. If the solution was feasible, the chromosome was not penalized. If the infeasible chromosome's fitness value was less than the population mean it was considered very poor solution and a fitness value 1.0 was assigned. If the fitness value exceeded the population average, then it was penalized by the infeasibility distance measure.

Their second technique was devised in Dammayer and Voss (1992). They used an ADD/DROP operator, which calculates the pseudo utilities according to criterions for handling more than one constraint.

The third technique is a filter operator and drops items from the knapsack until feasibility is reached. Items are selected randomly to maintain diversification.

The fourth and the most robust technique is the tabu operator. This operator is devised to improve either any randomly selected chromosome or the best chromosome in the population. With this operator, authors aim to avoid local optimums and provide improved chromosomes for GA. Since only feasible chromosomes are considered by the tabu operator, the filter operator is processed before the tabu operator to make the chromosomes feasible. The authors adopted the parameters defined in Dammayer and Voss (1993) for the tabu operator.

For GA parameters, the authors used one-point crossover, roulette wheel selection and flip mutation. The probability of crossover was set to 0.9 and the mutation probability was 0.009. Their initial population was random of size 50 but members were always fixed to achieve feasibility.

Their penalty operator yielded disappointing results, as the penalty function was too restrictive. The solution performance was improved by applying the ADD/DROP and Filter operators. A significant finding was that if the initial population was set up by an ADD/DROP heuristic, the simple GA was able to improve the performance of this heuristic by only 0.5% by using filter and ADD/ DROP operator.

The tabu operator proved to be the best operator. Optimal solutions were found for in most of the problems, but computing time increased. Selecting strings (solutions) based on the highest fitness value was better than randomly selected strings.

Another study that focuses on the parameterization of GA was by Hoff, *et al.* (1997). They started with recommended parameters from the literature but they did not cover all possible settings providing framework rather than a complete study. First,they used a population size of 50, one-point crossover, steady state generation with

replacement of the two worst chromosomes, random feasible initialization and 30,000 generations.

Inverted mutation was found to be most efficient with mutation probability 1/N, where N was equal to the chromosome length. They found that population size equal to 5*N yielded better solutions and burst crossover, a multi-point crossover was dominant in solution quality.

Having decided upon the parameters, 57 problems from the literature are solved using GA coded in C++. The average deviation from the optimum reported was 0.16% which is not significant. Hoff, *et al.* compared their findings with Theil and Voss and observed that the GA-TS operator algorithm performed slightly better than the random DROP/ADD operator of themselves.

3.2.2. Tabu Search and 0-1 MKP

Tabu Search (TS) developed by Glover in 1970 has proven robust and efficient in finding good solutions to optimization problems.

In TS, intensification guides the search into attractive regions. In contrast, diversification leads the search into new unexplored regions. Hanafi and Freville (1998) defined a new approach based on strategic oscillation and surrogate constraint information that provides a balance between intensification and diversification strategies.

Strategic oscillation is the frequency at which the critical levels are crossed in different directions. The authors define feasible solutions lying on or near the feasible region as critical levels. Those solutions that comply with this definition and stay infeasible are also considered critical levels. A promising zone is constructed by including all the critical solutions. Information deduced from the surrogate constraints is

used both in crossing and intensive exploration of the promising zone. These paths are formed by implementing constructive and destructive phases, depending on the current solution. In this context, the authors defined five different operators inheriting the characteristics of ADD and DROP heuristics and incorporating surrogate constraints to calculate pseudo utilities. Two aspiration criteria are applied in these operators. In the first one, when a move leads to new feasible solution better than the current best, the tabu status of the move is skipped. The second aspiration criterion is used if all the ones in the current solution are in tabu list when destructive heuristics are summoned.

The authors acknowledge that it is difficult to find a parameter setting at which the TS operates at optimum performance. The author's choice of oscillation parameters differ in intensification and diversification phases. In the intensification case, if the promising zone is reached from the feasible region, the search is focused on the feasible neighborhood of the current solution. Symmetrically, if the promising zone is reached from the infeasible side, the infeasible neighborhood of the current solution is searched. In diversification, the amplitude of the oscillation is given by the depth and near-feasible parameters that are user defined. On the infeasible side, the feasibility decision is studied with respect to three different constraint types: surrogate constraint, violated constraint and least saturated constraint. Along with the feasible side, these three constraint types are tested in order to observe the oscillation behavior. Findings have shown that the feasible side and infeasible side with surrogate constraint approach have a more regular oscillation than other constraints. Tabu status of a potential move is determined based on recency based and frequency based information.

Two different sets of problems are used to test the designed TS algorithm. The first set consists of 54 instances (Freville and Plateu, 1982) and optimal solutions are found for each of these problems. The second set is designed by Glover and Kochenberger (1995) and consists of 24 instances. Optimal solutions are not known for the last seven instances of this set.

Glover and Lokketangen (1998) described a TS approach for solving 0-1 Mixed Integer Programming (MIP) problems. Their study used two phases. In the first phase a basic, "first level", TS is explored. In the second phase, depending on the knowledge acquired from first phase, probabilistic deterministic measures for move selection and tabu tenure are explored.

Candidate list strategy is important to TS to guide the search to the next extreme point. In their algorithm, the trade-off between objective function value and integer infeasibility is used to select this element. In this context, two different approaches are used: a choice rule mechanism derived from surrogate constraints and a weighted sum of the two measures.

There are four types of defined moves in identifying a preferred extreme point;

I. Decreasing both integer infeasibility and profit.

II. Increasing both integer infeasibility and profit.

III. Nonincreasing integer infeasibility and nondecreasing profit.

IV. Decreasing profit and nondecreasing integer infeasibility.

The authors defined four different types of rules to evaluate these moves; weighted sum, ratio test, weighted sum but sorted within each group and ratio test but move type I and II. They study these rules in detail.

Recency and frequency based information are used to establish tabu status. Aspiration by integer infeasibility, by objective function value and by new best-detected solution are proposed aspiration criterion.

Probabilistic TS constitutes the second phase of this study. What is generally regarded as the Probabilistic TS (PTS) is usually applied to the move acceptance function. After tabu restrictions and aspiration criterion, PTS collects the evaluated moves in a candidate list and uses a biased probability to select from the list. The probability is biased to favor the better moves and this bias decreases exponentially. Two strategic oscillation schemes are recommended: strategic oscillation by parametric evaluation and by altered choice rules. Frequency based memory is used in diversification. They found that diversification depending on the inclusion of promising variables was found to be successful when compared with diversification depending on time spent in the basis.

Target analysis was used to efficiently identify the proper relationship between ratio test move parameters and to identify a better decision rule when diversification by penalizing time spent in the basis fails. In ratio test move analysis, they found that the relative ranking of move types I and II are important. Also, it is observed that changes in p, exponent component of integer feasibility for any particular solution, shifts moves to different move types.

The authors tested their findings with 57 problems from the literature. The problems are the same problems used in Chu and Beasley (1997), Theil and Voss (1993) , Hoff, *et al.* (1998), Hanafi and Freville(1998) and Drexl (1988). More on the characteristics of these problems is introduced in the next section.

Results of the experiments show that aspiration by objective function value levels worked well. This aspiration forces the moves toward the feasible region. For the first phase heuristic, the ratio test method, was found superior. For the second phase heuristic PTS, the threshold probability of move selection between 0.3 and 0.5 was best. Use of probabilistic measures for the move selection improved the average solution quality. In contrast, use of probabilistic tabu tenure yielded the same results as the first phase approach. PTS without the tabu memory converged to 'good' solutions the fastest but once that point was reached, this method could not improve the solution.

3.2.3. Simulated Annealing

Drexl's (1988) Probabilistic Exchange Algorithm (PROEXC) used simulated annealing (SA) to solve MKP. Simulated annealing is a random local search, allowing non-improving moves with probability t called the temperature. After r repetitions at temperature t, the temperature is reduced by a factor φ and repetitions are increased by a factor ρ. SA uses annealing of metalsas its basis. For initial cases, the author calculated t with

$$t = \alpha \beta \tag{3}$$

where $\alpha = \max \{c_j \mid \forall j\} - \min\{c_j \mid \forall j\}$

and found that PROEXC works best at $\beta = 0.5$, $\varphi = 0.6$. In all these cases, r and ρ were equal to the number of variables and 1.1, respectively. Later experiments found n was a good value for r, but 1.2 better value for ρ.

The author tested PROEXC with 57 problems from the literature. PROEXC's overall computation time is reported as very fast when compared to other heuristics, with good solution quality.

3.3. Comparison of Heuristics.

When a new heuristic is published in the computational and mathematical sciences literature, its contributions should be evaluated scientifically and reported in an objective manner (Barr, *et al.*, 1995). In this context, many new heuristics are tested by using standard or synthetic test problems.

Zanakis (1977) examines the performance of three heuristics (Senju-Toyoda, Kochenberger *et al.* and Hillier) applied to 0-1 integer programming problems. Using a designed experiment, three factors were considered: number of variables (15,30,45), number of constraints (10,20,30) and degree of constraint slackness (0.3,0.5,0.9). CPU time, error and relative error were measured for each problem. Synthetic test problems were generated randomly from a uniform distribution and 5 replications were used. Analysis of variance and stepwise regression were used to study the effects of these factors.

Hill and Reilly (1997) investigate the effects of correlation structure and constraint slackness settings on the performance of solution procedures on synthetic two dimensional knapsack problems. They investigated how the performance of branch-and-bound (CPLEX) techniques and Toyoda's heuristic are affected by problem structure. A total of 45 feasible correlation structures were examined. The size of the problems was fixed to be 100 variables and each constraint's slackness was set to the values 0.3 and

0.7. They used the Pearson product-moment correlation induction method and the Spearman rank correlation-based correlation induction method in defining correlation between the objective function and constraint coefficients. Depending on these parameters, they produced five random problems for each point of their factorial design generating 2240 problems. Two non-parametric statistical tests are used to analyze the data from the experiment. They tested correlation structure influence, individual correlation term influence, constraint slackness influence and interaction of the last two. They found that problem correlation structure affects the solution quality

Lokkatengen (1997) compares TS and GA performance on solutions of MKP. The GA algorithm (Hoff *et al.* 1997) and TS algorithm (Lokkatengen and Glover, 1997) , reviewed above were used to solve the 57 standard test problems. Each algorithm was set up with the findings of past studies. The author compared these two algorithms by how many times they found the optimum solutions for test problems. The results favored the GA since it was able to find 56 out of 57optimal solutions. The basic TS performed poorly and was able to find only 39 optimal solutions. However, when the more advanced TS mechanisms of strategic oscillation, diversification and intensification were applied, all of the problems were solved within 20*N iterations. Thus, Lokkatengen recommends a hybrid algorithm as proposed by Theil and Voss (1993).

Hanafi, et al. (1996) compared variants of the Simple Multistage Algorithm (SMA) and variants of TS algorithms. SMA incorporates different local search strategies in a "flexible fashion." SMA starts with diversification by generating random solutions in addition to primal and dual solutions. These solutions were used as initial start points for neighborhood searches. Feasibility was maintained during the whole process by

projecting infeasible solutions into the feasible domain by using the Senju and Toyoda's (1968) dual algorithm.

Three local search algorithms were used in SMA. The first one was SA described in Drexl (1988). The second local search, Threshold Accepting (TA), accepts any solution within an infeasible threshold. The third technique is a relatively new concept that introduces noise to the data to overcome local optimality. Noising Method (NM) introduces noise into solutions to diversify the search, reducing the noise introduced as the search converges.

The TS algorithms compared were: Reactive Tabu Search (RTS), Reverse Elimination Method (REM), TS using Balas and Martin (1980) as a subroutine, Critical Event TS, and two other TS approaches.

The algorithms above were tested with 54 standard problems. Test of SMA with different local search approaches and AGNES algorithm of Freville and Plateau (1990) proved that SMA worked best with TA. The solution quality for SMA was not better than any other algorithms in the literature. However authors defined their SMA as simple to implement and fast in solution. AGNES solved most of the problems at their optimums and was the best when compared with SMA applications. TS comparisons showed that TS defined in Glover and Kochenberger (1995) and the infeasible version of the author's TS were the best as they found optimal solutions for each of the problems.

3.4. Test Problems.

The effectiveness of any proposed methodology for solving a given class of problems can be demonstrated by theoretical analysis and empirical studies (Barr, *et al.* 1995). Analytical studies may not always be possible (Hooker, 1994). In this case,

empirical studies are the only tools to assess the effectiveness of an algorithm. Empirical studies can be conducted on either real-world problems or synthetic test problems. Synthetic test problems generally may not resemble the real-world problems, they simulate (Hooker, 1995). However, synthetic test problems can fully vary the problem parameters yielding more information about algorithm performance.

Computational experiments with algorithms are usually undertaken (1) to compare the performance of different algorithms for the same class of problems or (2) to characterize or describe an algorithm's performance (Barr *et al.* 1995). As in many other sciences, error and variation may be present in computational experiments. Hill (1998) addresses one of these errors as "oversight error". This error occurs when a potentially significant factor is missed in testing. Since correlation structure affects solution procedure performance, the unaccounted effect of correlation in a test problem can potentially bias analytical results. Hill (1998) examines the correlation structure of standard MKP problems and believes the structure may in fact influence the solution procedure performance.

4. GA FOR MKP

4.1. Why one more paper on parameterization?

There is not any theorem that explains why GA`s have the characteristics that they have (Beasley, *et al.*, 1993). Thus, we can not analytically predict which parameter settings are appropriate for a particular problem set. Schaffer, *et al.* (1994) has shown that the optimal parameter settings in GAs differ with the problem type solved. In fact, the time and resources required to find optimal parameters for a problem domain are

often orders of magnitude greater than the time one plans to spend solving problems in the domain (Davis, 1991). For these reasons, many papers seek robust parameter settings that work for a class of problems. This study seeks robust settings for problems that explicitly vary correlation structure.

4.2. GA Operators and Parameters for 0-1 MKP

4.2.1. Representation.

The most intuitive way to represent a 0-1optimization problem is using binary representation and setting a variable to its corresponding gene alele, either 0 or 1. Theil and Voss (1993) found this representation better than alternative representations.

4.2.2. Initialization.

Initial GA population may be seeded or random. Since population seeding may cause early convergence to local optimal solutions, we generated the initial population randomly. Based on Hill and Reilly's (1997) study, we set at 35% the probability used to create the initial population and corrected infeasible solutions. Primal and dual initialization techniques were examined and discontinued as not promising because of slow convergence and poor solution quality

4.2.3. Scaling.

Scaling has two important features in GA. First, it prevents the dominance of super individuals in the early stages of evolution. This feature helps to overcome premature convergence, which is the least desired characteristic in an optimization heuristic. Second, scaling helps as the average fitness of population approaches the population maximum, when selection schemes may bog down. Maximum score and the average score may have the same chance of being selected. In this case, scaling may help.

Initial tests compared three different scaling schemes, and the no scaling option. Signed rank test statistics showed that sigma truncation scaling provided better solution quality. The formula for sigma truncation scaling is:

$$F = \text{obj\_value} - (\text{obj\_ave} - c * \text{obj\_dev}) \text{ where c is } 2; \quad (4)$$

4.2.4. Fitness Function.

A fitness function must incorporate constraints into the function to reflect the feasibility of chromosomes. There are three ways to handle infeasibility. First, infeasible solutions can be penalized, reducing the attractiveness of infeasible chromosomes. Second, repair operators can transform infeasible solutions, into feasible ones. Third, any infeasible chromosome can be killed and reproduction repeated until feasible solutions remain.

In pilot studies, we implemented the penalty functions available in the literature and found that they were not efficient for MKP. We then designed a penalty function that considers the distance ratio of feasibility to the current solution. After numerous pilot runs, this penalty function did not seem promising either.

We then devised and employed a repair operator, which randomly starts from a loci and drops items until feasibility is reached. The algorithm then tries to add as many items back as possible before violating any constraint. This strategy was devised to avoid the epistasis problem. Discussions in Hoff, et al. (1998), Theil and Voss (1993) and Michalewicz (1992) can be related to epistasis.

4.2.5. Selection.

Exploration and exploitation are two phases that an algorithm uses to reach a global optimum. Exploration guides the search into unvisited parts of the search space

while exploitation helps the search to remember the knowledge acquired in previous visits to find better solutions. The balance between these two phases plays an important role in tuning the search to find global optimality. GAs possess both phases. Whitley (1994) reports that there are two important factors in GA: population diversity and selective pressure. He advises increasing the selective pressure to cause exploitation and decreasing it to cause exploration.

In our implementation a tournament selection scheme applies more selective pressure than does roulette wheel selection. Therefore, we tested both of them to learn about the effects of selective pressure in solution quality.

4.2.6. Crossover.

The empirical results are divided on the best type of crossover and the best value for the probability of crossover.

Based on these discussions, we included crossover type in our experimental design, specifically uniform and two-point crossovers. The probability of crossover values of 0.85 and 0.95 are also included in the test design.

4.2.7. Population Size

Population size is one of the most important parameters in the GA. Population size influences both population diversity and selection pressure. A small population size may result in premature convergence; if it is large, computing time may be wasted without any significant return. Trial runs suggested we employ the Theil and Voss (1993) recommended population size of between 50 and 100.

4.2.8. Mutation and Probability of Mutation.

Mutation operator helps diversify the population and search. Schaffer, *et al.* (1989) found that the mutation rate is sensitive to population size. Based on our population size, we tested probability of mutation between 0.01 and 0.03.

4.2.9. Steady State or Generational Replacement

In each generation, some individuals are excluded from the population while offspring are included. The ratio at which this replacement takes place defines the type of GA. If the number of replacements is just one or two, this type of GA is called Steady State GA (SSGA). If the whole population is replaced as recommended by Goldberg (1989), it is called Generational Replacement GA.

The advantages of SSGA include (1) schema fitness versus percentage in the population works out properly as the fixed point of the system; (2) good members of the population float to the top of list where they remain and (3) poor individuals leave the population (Syswerda, 1994). We decided to test SSGA with the replacement of 25 individuals in each generation.

4.3. Coding

Galib© version 2.4.3 by Wall (1998) was used to implement our GA. We used binary string genome (chromosome) to implement the GA and changed some of the statistics objects to fit our analysis requirements.

4.4. Experimental Design.

There were six factors assumed significant in GA performance. These are crossover type, selection scheme, population size, probability of crossover, probability of mutation and number replaced. For six factors, a full factorial design requires 64 runs, which is computationally very expensive. We used a fractional factorial design to screen out effects and selected 2^{6-1}_{VI} fractional factorial requiring 32 runs (Table1).

The test problem set of Hill and Reilly (1997) includes optimal or best-known integer solutions so GA solution quality measured as a relative error was the primary measures of effectiveness. The formula for relative error is given below.

Z_{IP} : Optimum or best known solution

Z_{CUR} : Current solution

$$REL := 100 \cdot \frac{\left(Z_{IP} - Z_{CUR}\right)}{Z_{IP}} \tag{5}$$

GAs actually offer two performance measures of solution quality: online average and offline average. Online average is the average performance of all structures tested during the search. Offlineaverage uses the best structure value for each evaluation in the average. Online average penalizes the search if poor solutions are created by operators, where offline average does not.

Table 1 2^{6-1}_{IV} Fractional Factorial Design

Design Setting	Crossover	Selection	Population Size	Prob. Crossover	Prob. Mutation	Number Replaced
1	Uniform	Tournament	50	0.85	0.01	2
2	Two-point	Tournament	50	0.85	0.01	25
3	Uniform	Roulette	50	0.85	0.01	25
4	Two-point	Roulette	50	0.85	0.01	2
5	Uniform	Tournament	100	0.85	0.01	25
6	Two-point	Tournament	100	0.85	0.01	2
7	Uniform	Roulette	100	0.85	0.01	2
8	Two-point	Roulette	100	0.85	0.01	25
9	Uniform	Tournament	50	0.95	0.01	25
10	Two-point	Tournament	50	0.95	0.01	2
11	Uniform	Roulette	50	0.95	0.01	2
12	Two-point	Roulette	50	0.95	0.01	25
13	Uniform	Tournament	100	0.95	0.01	2
14	Two-point	Tournament	100	0.95	0.01	25
15	Uniform	Roulette	100	0.95	0.01	25
16	Two-point	Roulette	100	0.95	0.01	2
17	Uniform	Tournament	50	0.85	0.03	25
18	Two-point	Tournament	50	0.85	0.03	2
19	Uniform	Roulette	50	0.85	0.03	2
20	Two-point	Roulette	50	0.85	0.03	25
21	Uniform	Tournament	100	0.85	0.03	2
22	Two-point	Tournament	100	0.85	0.03	25
23	Uniform	Roulette	100	0.85	0.03	25
24	Two-point	Roulette	100	0.85	0.03	2
25	Uniform	Tournament	50	0.95	0.03	2
26	Two-point	Tournament	50	0.95	0.03	25
27	Uniform	Roulette	50	0.95	0.03	25
28	Two-point	Roulette	50	0.95	0.03	2
29	Uniform	Tournament	100	0.95	0.03	25
30	Two-point	Tournament	100	0.95	0.03	2
31	Uniform	Roulette	100	0.95	0.03	2
32	Two-point	Roulette	100	0.95	0.03	25

5. TEST RESULTS

5.1. Penalty Function Versus Repair Operator

We devised and tested a penalty function penalizing infeasible solutions according to proportion of constraint violation. We found it difficult to define one penalty function adequate for all test problems in the data set. We then tested a repair operator. Figure4 is a comparison of repair operator and penalty function performance. Figure 4 compares performance on one test problem but represents the repair operator dominance observed in all cases. The repair operator is used in the rest of the analysis.

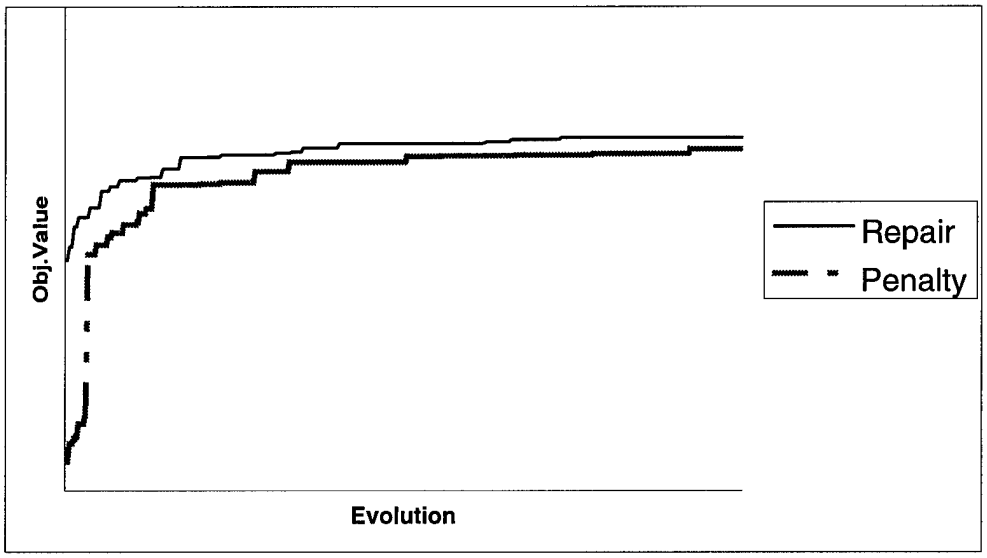


Figure 4. Repair and Penalty Methods (Problem 665)

Since a GA is stochastic, each problem is solved five times, for each experimental design setting. The GA ran for 5000 generations. This value was selected based on trial run experience. Since optimal values for some problems were unknown, termination

before 5000 generations was not considered. Our results improved the best known values for the four test problems without a known optimal solution value. This is summarized in Table 2.

Table 2 Improved Solutions

Problem Number	LP Solution	Best Known	GA
1027	1731	1724	1725
1029	3859	3849	3850
1061	2562.7	3549	3551
1116	3868.3	3865	3866

5.2. Robust Parameters for GA

As seen in Table 3 the most robust design setting was design point 31 (uniform, roulette-wheel, 100, 0.95, 0.03, 2). The overall relative error was 0.0239% with standard deviation 0.0599%. Further, the optimal solution was found for 848 of 1120 problems (76%). Figure 5 plots the mean relative error and 95% confidence bound for the 32 design points. The main effect which changes at design point 16 was probability of mutation. It was observed that relative error was significantly reduced from this point on. Complying with this observation, we tested probability of mutation values 0.04 and 0.06 and found reduced solution quality for number of generation 5000.

Design setting 19 (uniform, roulette wheel, 50, 0.95, 0.03, 2) performed best within the settings with population size 50. Student's *t* test, Tukey's Multiple Comparison tests and Ranking and Selection concluded that design settings 31 and 19 are different significantly. More details on comparisons can be found in Appendix B.

Next, we analyzed the data by using statistical tools to learn more about the effects of GA parameters on solution quality. Normality and constant variance

assumptions were checked and found that they were not met. Thus, we transformed the data by using square root and natural logarithm and ran a stepwise regression. The resulting effect plots are in shown Figure 6.Deviation from a level plot indicate a potentially significant effect. The number of replacements and probability of crossover effects were insignificant. However, the number of replacements drives up computation time. Despite its longer computation time, replacing 25 offspring in each iteration was not as good as replacing 2 at a time. Thus replacing 2 at a time is best in both computation time and solution quality.

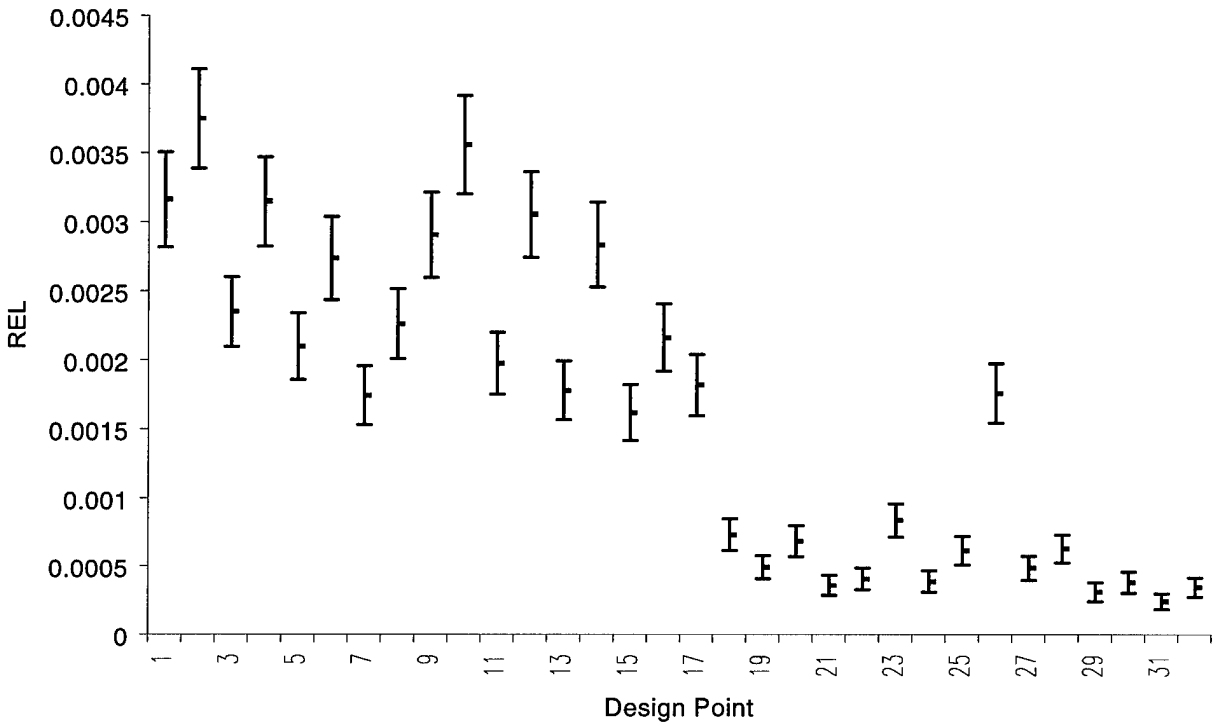


Figure 5. Confidence Intervals For REL

Uniform crossover, roulette wheel selection, 0.03 probability of mutation and population size 100 performed better than their rival settings. Selection plot favored roulette-wheel selection, which does not have the selective pressure of tournament

selection. This choice supported exploration more than exploitation. 0.03 probability of mutation was better than 0.01.

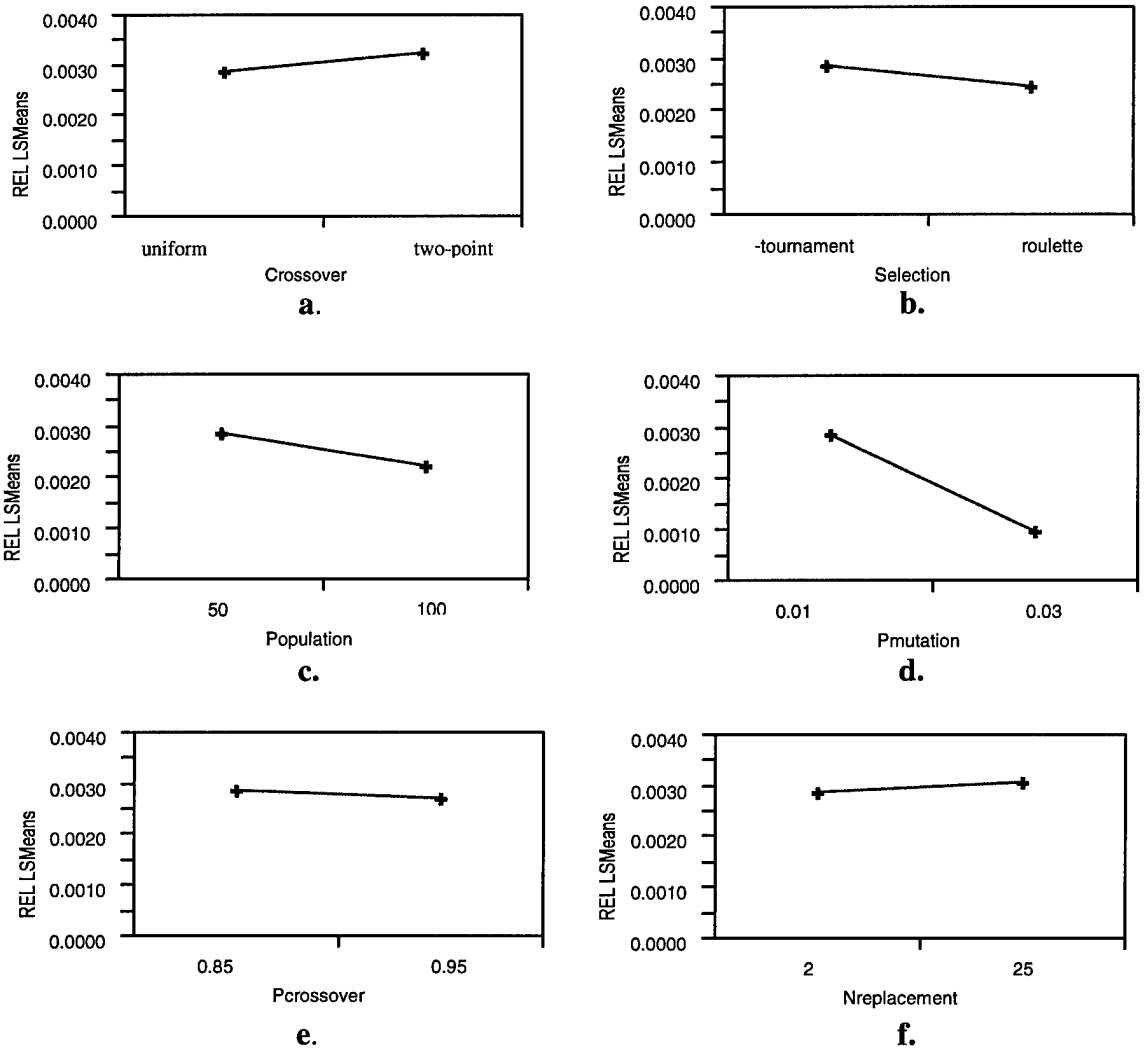


Figure 6. Effects of GA Parameters on Solution Quality

For further investigation of robust parameters, we tested probability of mutation values of 0.04 and 0.06, and population size 30. Results were evidence of the fact that further increasing probability of mutation degrades the solution quality. Decreasing population size did not help GA performance either.

Table 3 Test Results

Design Setting	Crossover	Selection	Pop. Size	Prob. Crossover	Prob. Mutation	Number Replaced	Mean REL	Number of Optimal
1	Uniform	Tournament	50	0.85	0.01	2	0.003158	221
2	Two-point	Tournament	50	0.85	0.01	25	0.003749	184
3	Uniform	Roulette	50	0.85	0.01	25	0.002359	291
4	Two-point	Roulette	50	0.85	0.01	2	0.003213	212
5	Uniform	Tournament	100	0.85	0.01	25	0.002114	341
6	Two-point	Tournament	100	0.85	0.01	2	0.002826	260
7	Uniform	Roulette	100	0.85	0.01	2	0.00154	328
8	Two-point	Roulette	100	0.85	0.01	25	0.002282	307
9	Uniform	Tournament	50	0.95	0.01	25	0.002875	244
10	Two-point	Tournament	50	0.95	0.01	2	0.003573	190
11	Uniform	Roulette	50	0.95	0.01	2	0.001993	332
12	Two-point	Roulette	50	0.95	0.01	25	0.00306	230
13	Uniform	Tournament	100	0.95	0.01	2	0.001798	369
14	Two-point	Tournament	100	0.95	0.01	25	0.002819	245
15	Uniform	Roulette	100	0.95	0.01	25	0.001313	498
16	Two-point	Roulette	100	0.95	0.01	2	0.002141	324
17	Uniform	Tournament	50	0.85	0.03	25	0.001087	623
18	Two-point	Tournament	50	0.85	0.03	2	0.000723	593
19	Uniform	Roulette	50	0.85	0.03	2	0.000509	674
20	Two-point	Roulette	50	0.85	0.03	25	0.000662	596
21	Uniform	Tournament	100	0.85	0.03	2	0.000366	779
22	Two-point	Tournament	100	0.85	0.03	25	0.00041	736
23	Uniform	Roulette	100	0.85	0.03	25	0.001194	825
24	Two-point	Roulette	100	0.85	0.03	2	0.000369	748
25	Uniform	Tournament	50	0.95	0.03	2	0.000595	638
26	Two-point	Tournament	50	0.95	0.03	25	0.00112	601
27	Uniform	Roulette	50	0.95	0.03	25	0.00048	694
28	Two-point	Roulette	50	0.95	0.03	2	0.000635	608
29	Uniform	Tournament	100	0.95	0.03	25	0.000317	810
30	Two-point	Tournament	100	0.95	0.03	2	0.000394	774
31	Uniform	Roulette	100	0.95	0.03	2	0.000239	848
32	Two-point	Roulette	100	0.95	0.03	25	0.000822	772

5.3. Convergence

Our next concern was to examine the convergence trend of attractive parameter settings. Design settings 31 (uniform, roulette wheel, 0.95, 0.03, 100, 2) and 19 (uniform, roulette wheel, 0.85, 0.03, 50, 2) offer the best solution qualities but they differ in average computation time (approximately 3.8 and 5.2 sec respectively). Investigating their convergence trend can give us some clues about the trade-off that we can give up between solution quality and computation time. Furthermore, we have observed that the parameter settings with 0.01 probability of mutation were worse in solution quality. At this point we can suspect the premature convergence and a convergence graph may reveal the facts about their inefficiency. Thus, we included design setting 11 (uniform, roulette wheel, 0.95, 0.01, 50, 2) which is the best among settings with probability of mutation 0.01. We have chosen six problem instances, which favor different combinations of these three GA parameter settings. Figure 7-12 display the convergence trend of three parameter settings.

Our initial populations are very good: all solutions are feasible with many items in the knapsack. We observe that less diversification causes converge to near optimal solutions quicker. However, once a solution close to optimal is reached, improvement disappears. This may be evidence of the fact that exploration halted due to less probability of mutation in a small population. With increased diversification, convergence is slower initially but continue to an improved final solution

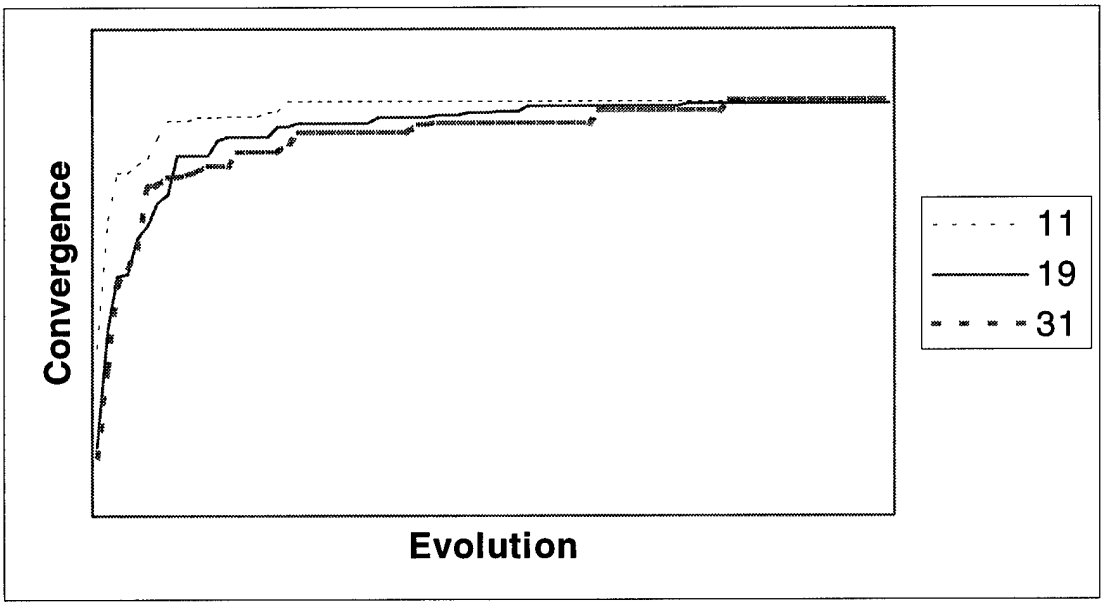


Figure 7 Problem 544 (0, 0, 0, 0.3, 0.3)

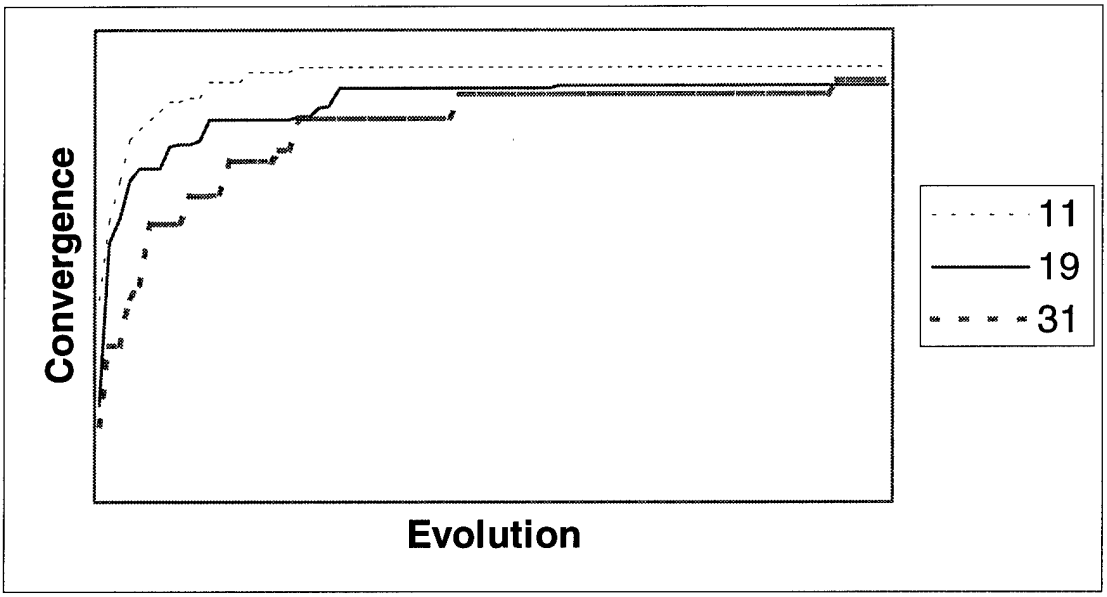


Figure 8 Problem 31 (0.49887, 0.49887, 0.99752, 0.7, 0.3)

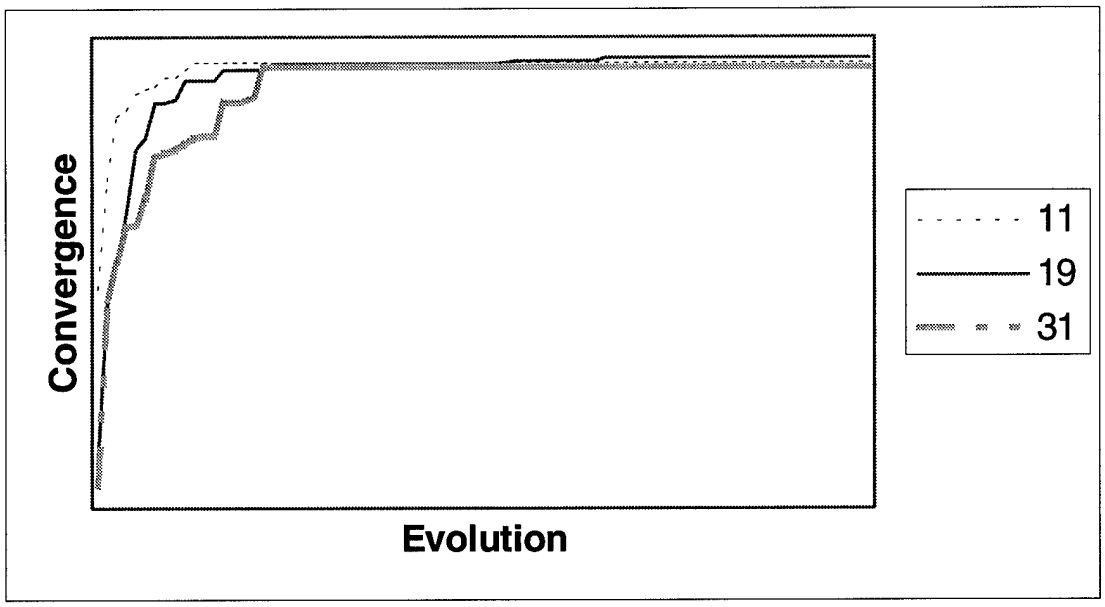


Figure 9 Problem 746 (-0.49887, 0.99752, -049887, 0.3, 0.7)

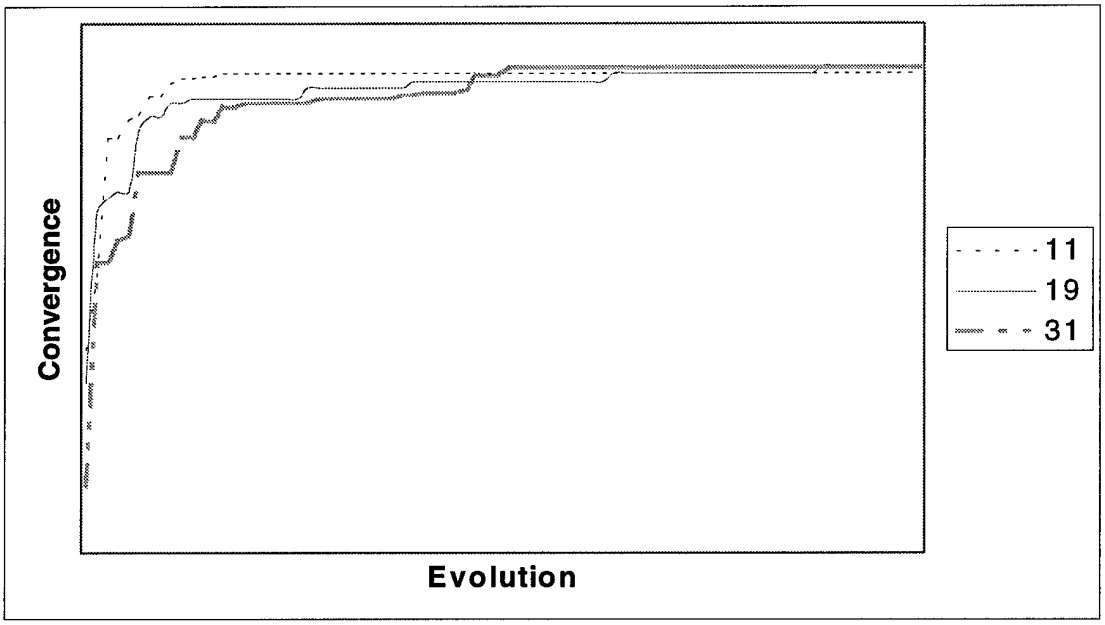


Figure 10 Problem 171 (0.49887, 0.49887, 0.49887, 0.7, 0.3)

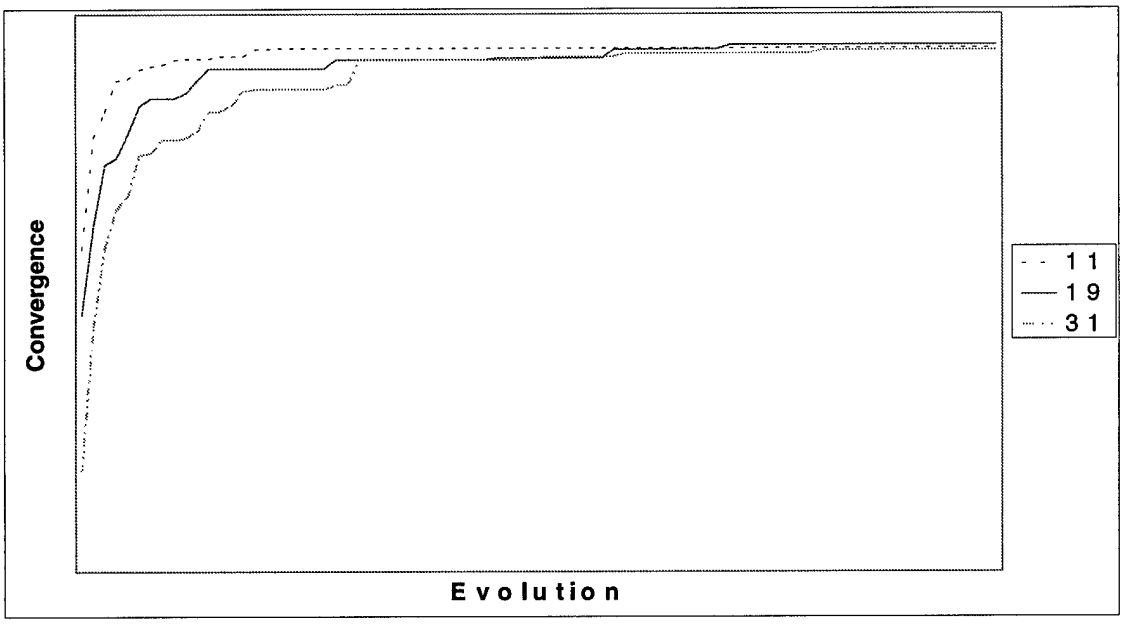


Figure 11 Problem 254 (0, 0, 0.49887, 0.7, 0.3)

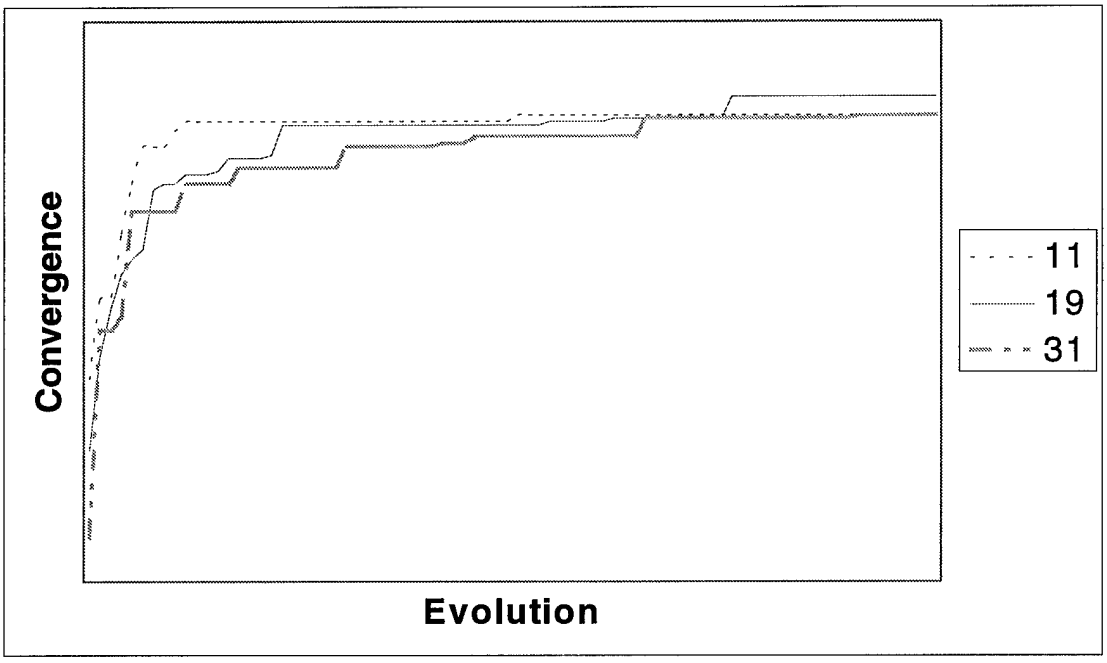


Figure 12 Problem 121 (-0.49887, -0.49887, 0.99752, 0.3, 0.3)

Competition between design settings 19 and 31 re-visits in the plots above. In early evolution ages, population with small size has more tendencies to reach optimal quicker than population with bigger size. After some point in the evolution bigger population sized GA catches small sized and it is hard to describe which one is attractive from this point on. So, we can again conclude that design point 19 is more favorable due to less computation time.

5.4. Effects of Problem Structure on Solution Quality

To investigate the effects of problem structure, we selected two GA parameter settings: the most robust setting (31) and the least robust setting (2). Then we solved Hill and Reilly's (1997) problem set and averaged the five relative error values for each problem structure setting. Hence, we had response values for all 224 problem structures.

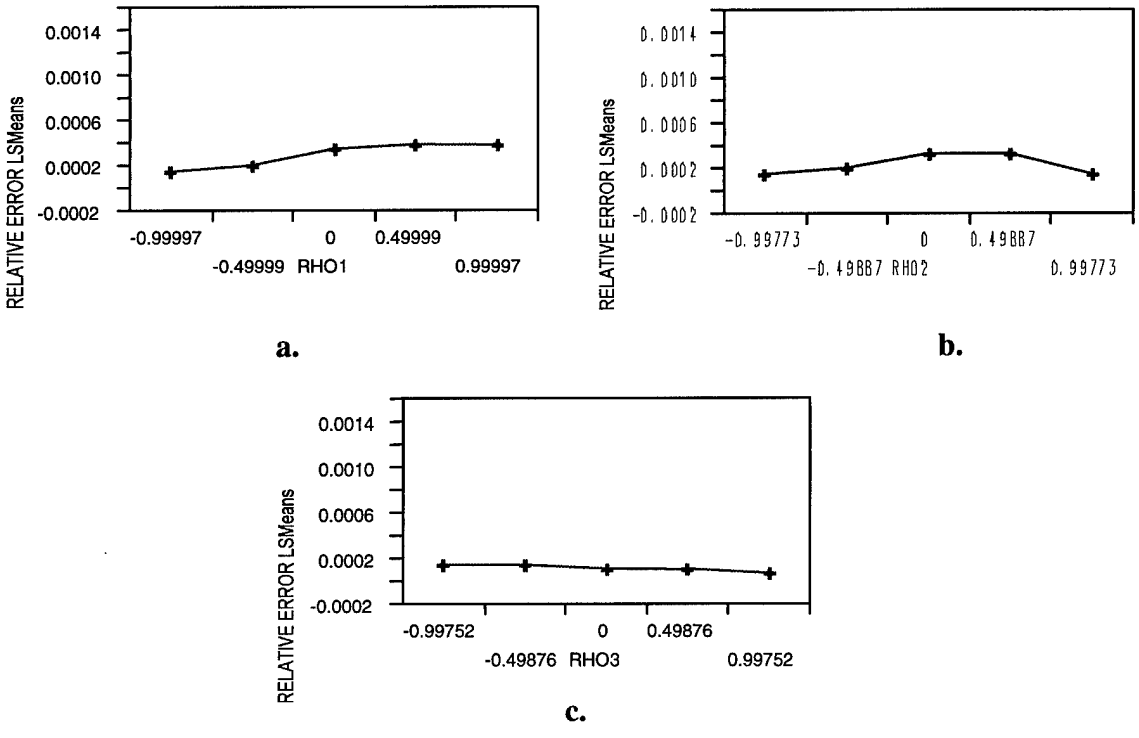
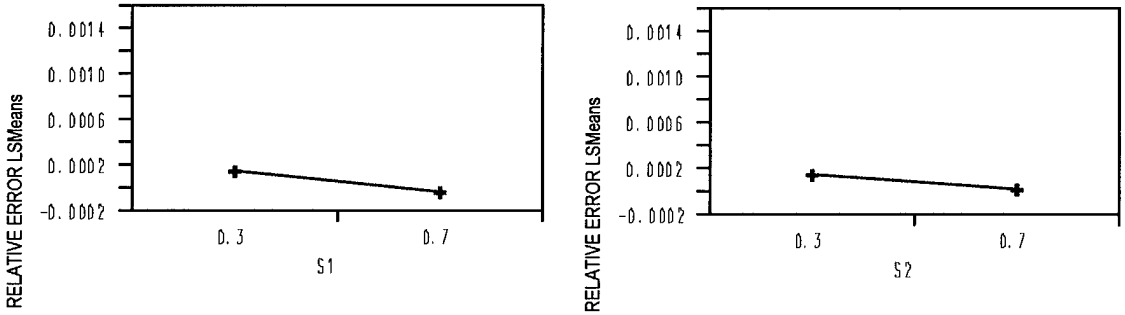


Figure 13 Correlation Structure Effects

Figures 13a-13c show the effect of correlation on solution quality. For Figures 13a-and 13b, correlation between objective function and first constraint, and correlation between objective function and second constraint respectively, these effects are statistically significant at 95% confidence ($\alpha = 0.05$). Solution quality starts to degenerate as the correlation gets greater than 0 and either stays there at 0.499 or drops afterward. Thus, we can conclude that regardless of GA parameters used, the performance of the GA is affected by correlation among objective function coefficients and the constraint coefficients and appears worst within a correlation range of 0.0 and 0.5.

Hill (1998) found the test set of Chu and Beasley (1998) have correlation values in this 0.0 to 0.5 range. However, our problems have only two constraints while their problems vary the number of constraints among 5,10 and 30. Resolving the Beasley and Chu test set with our GA is worthy of research.

Figures 14a and 14b display the constraint slackness effect on GA performance. Both constraint slackness effects, S1 and S2, are statistically significant and as in many other heuristics, GA favors higher slackness value. As in many past studies, tighter constraints yielded harder problems.



a. **b.**

Figure 14 Slackness Effect

6. CONCLUSION

In this paper we have studied the effects of problem structures to find the most robust parameter settings for GAs used to solve the MKP. Our study differed from previous ones in the way that we repaired infeasible chromosomes with a random operator and examined a different synthetic test problem set.

A random repair operator is efficient in the sense that it does not require calculation or exploitation of any problem specific knowledge. Moreover, this operator can easily be implemented and computing time consumed by this operator is minimal.

The most confusing part of analysis was in terminating the GA. Our default termination counted generations and did not take number of evaluations into account. Actually this confusion revealed an important experience in our research. Under this termination rule apparently the design with 25 replacements had more chances to produce and evaluate than 2 replacements. The expected result would be to have design points with 25 replacements dominate the others. However, our first macro runs showed otherwise and we were firmly satisfied with that replacing 2 chromosomes is better for the MKP. In addition results of our alternative termination rule, which terminates after certain number of evaluations and compensates for different replacements numbers, validated our findings about GA type.

The most robust design point was 31 and ranking and selection, confidence interval, Mann -Whitney rank-sum and Student's *t* tests validated this result. This setting includes population size 100. As we discussed earlier, increasing population size may

cause wasting computing time without any significant return. Non-parametric rank sum tests have shown that design points 31 and 19, best with population size 100 and 50, respectively, were statistically different. Furthermore, computation times are increased approximately 1.5 times when population size is 100. Henceforth, if we notice that the solution quality of both design points is practically insignificant, design point 19 with probability of crossover increased to 0.95 may be more attractive in cases where computation time is a critical concern.

As with any empirical study our findings will not always be true. However, we applied a systematic approach on test problem instances with a range of correlation structures and slackness settings.

There are other avenues to extend this work. For instance, we used a random repair operator. Another study might examine purely random populations with penalty functions. Future studies should examine problems with varied correlation structures and more constraints, and compare these to solutions on the Beasley and Chu data set. Finally, another study might re-examine the GA parameters proposed in the literature with those proposed here.

Chapter 3. Future Avenues

This thesis shed a light on the valid empirical parameterization of GA with a special repair operator. Test problems were diversified enough to cover all possible correlation and slackness settings in 0-1 Bi-dimensional Knapack Problem.

There are a few things which need to be done to complete the findings of this thesis. One can implement and study a hybrid operator in GA which projects infeasible solutions onto the feasible region in a systematic way. Findings of such study can be compared with the research presented here to learn more about the behavior's of operators

Another research may investigate the effect of problem size on GA performance. Our study focused on two constraints. A real world application of MKP might have more constraints. This kind of research may provide the information about trends in GA when compared with ours.

Finally, another open and interesting research area may exist in creating test problems which counts for "epistasis". Correlation measures the relations between rows and is found to have significant effect on solution quality. If one can achieve to create test problems which varies interaction degrees of variables and keeps a targeted correlation structure, we can be more confident about the findings of any study especially related to GAs.

APPENDIX.A

Appendix A.1 Heuristics And Reduction Methods For Multiple Constraints 0-1 Linear Programming Problems

Freville and Plateu (1986) described an "automatic code" which reduces the size of any multiple inequality constraint 0-1 linear programming problem. Upon this context, the authors defined two greedy algorithms (AGNES I and AGNES II) that determined lower bounds for the optimal solutions by bringing into play the concepts of surrogate relaxation, oscillating assignment and strongly determined variables. These algorithms are inserted into a reduction frame which reduced the constraints and variables with the help of Lagrangean and surrogate relaxation.

In the first phase of their algorithm, three methods were followed to calculate surrogate multipliers: The surrogate problem and its relaxation was then solved by the NKR and FPK 79 algorithms of Plateau. The solution produced by the NKR algorithm was fixed by AGNES I and AGNES II which differs in the way that they assign 0 or 1 value to some free variables. After these value assignments, a simple procedure was followed to reduce the size of current problems in order to generate a well-stated problem.

The authors tested their methods with 20 standard problems from the literature and 30 random test problem generated by themselves. They used randomly generated test problems to compare their algorithm with the other well-known algorithms. Results of their tests shown that AGNES' accuracy is at least as good as Balas and Martin (1980) but with increased computation time.

Authors compared efficiency of their reduction algorithm by solving reduced problems with Shih's (1979) algorithm and concluded that as the problem size gets larger, their reduction technique promised less computation time. Exclusively, they reported an algorithm which bounds on the sum of the number of variables equal to 1 at optimum.

Appendix A.2 A Heuristic For General Integer Programming

Kochenberger, et al (1974) devised an algorithm, which adopted the 0-1 algorithm of Senju and Toyoda to solve general integer programming problems. They started with all variables equal to zero and added items one at time according to the amount of objective function increase they would produce. Their algorithm did not restrict variables to be binary. Problem coefficients were not restricted to be nonnegative. However, the authors did not guarantee the feasibility of solutions when the problems with the negative cioefficients would be solved with this algorithm. The authors generated some test problems which proved this fact about negative coefficients and recommended backtracking in case of infeasible solutions.

Their algorithm can be applied to 0-1 problems by introducing constraints which forbid the procedure from introducing more than one unit.

Appendix A.3 An Approximate Algorithm For Multidimensional Zero-One Knapsack Problems - A Parametric Approach

Lee and Guignard (1988) proposed an approximate algorithm consisting of finding a feasible solution with Toyodas'(1975) primal algorithm, fixing variables and complementing certain variables with triple complementing technique of Balas and Martin (1980). Proposed modification to Toyoda's primal method consisted of selecting as many variables as possible at each iteration, as long as feasibility is guaranteed, instead of selecting one at a time. Although solution quality of Toyoda's technique was better, modification reduced the computation time significantly.

The authors' procedure embodied two phases. PHASE I implements a modified version of Toyoda's primal technique to find a good feasible solution. Then problem size is reduced by fixing a certain set of variables using the reduced costs from LP relaxation. The LP relaxation of the problem is solved using the simplex algorithm. The simplex algorithm was seeded by using the variables from the modified Toyoda algorithm. Phase II aims to improve the solutions derived in PHASE I by the triple complementing procedure defined in Balas and Martin (1980). In triple complementing, complements of a triplet of variables (one at 0, one at 1 and one at 0 or1) is searched to find better solutions.

Their procedure comprises three parameters: k,r,p. Parameter k is the trade of between solution quality and computation time in PHASE I. Parameter r determines if the complementing must be carried out in a certain iterations, because complementing was

computationally expensive and not promising most of the time. Finally parameer p influences the trade-off between solution quality and time in PHASE II.

The authors tested their procedure with 48 problems. They reported that their algorithm yields better solutions than Toyoda (1975) and Magazine and Oguz (1984) but it was outperformed by Balas and Martin (1980).

Appendix A.4 A Heuristics Algorithm For Multidimensional Knapsack Problem.

Magazine and Oguz(1984) devised an approximate algorithm which combines the techniques in Senju and Toyoda (1968) with Everett's Generalized Lagrange (EGL) multiplier approach. Authors found LP relaxation solution of the problems useful as they represent a bound for the possible solution. All the variables are set to 1 at the start in their dual algorithm. Then feasibility was reached by dropping the variables using some greedy rules. These rules included one coefficient, which is perturbed until finding only one variable that violates one constraint. Then this variable was set to 0.

They produced 75 random problems to test their algorithm. In these problems, objective function and constraint coefficients were derived from uniform distribution. Both variable size and constraint size varied from 20 to 1000.

They reported that their algortihm performed sligthly worse than the algorithm of Kochenberger, et al (1974) in solution quality. However, their algorithm performed remarkably better in terms of computation time.

APPENDIX B

Appendix B.1 Comparison Of Design Settings With Student's t Test. Positive values show pairs of means that are significantly different.

Table 1 Student's *t* Test

Design Point	2	10	4	1	12	9	6	14	3	8	16	5	11	13	7	15
2	-0.0001	0.0000	0.0004	0.0005	0.0006	0.0007	0.0008	0.0008	0.0013	0.0013	0.0015	0.0015	0.0016	0.0018	0.0021	0.0023
10	0.0000	-0.0001	0.0002	0.0003	0.0004	0.0006	0.0006	0.0006	0.0011	0.0012	0.0013	0.0013	0.0014	0.0016	0.0019	0.0021
4	0.0004	0.0002	-0.0001	-0.0001	0.0000	0.0002	0.0003	0.0003	0.0007	0.0008	0.0009	0.0010	0.0011	0.0013	0.0015	0.0018
1	0.0005	0.0003	-0.0001	-0.0001	0.0000	0.0001	0.0002	0.0002	0.0007	0.0007	0.0009	0.0009	0.0010	0.0012	0.0015	0.0017
12	0.0006	0.0004	0.0000	0.0000	-0.0001	0.0001	0.0001	0.0001	0.0006	0.0006	0.0008	0.0008	0.0009	0.0011	0.0014	0.0016
9	0.0007	0.0006	0.0002	0.0001	0.0001	-0.0001	-0.0001	-0.0001	0.0004	0.0005	0.0006	0.0006	0.0007	0.0009	0.0012	0.0014
6	0.0008	0.0006	0.0003	0.0002	0.0001	-0.0001	-0.0001	-0.0001	0.0003	0.0004	0.0006	0.0006	0.0007	0.0009	0.0012	0.0014
14	0.0008	0.0006	0.0003	0.0002	0.0001	-0.0001	-0.0001	-0.0001	0.0003	0.0004	0.0005	0.0006	0.0007	0.0009	0.0011	0.0014
3	0.0013	0.0011	0.0007	0.0007	0.0006	0.0004	0.0003	0.0003	-0.0001	-0.0001	0.0001	0.0001	0.0002	0.0004	0.0007	0.0009
8	0.0013	0.0012	0.0008	0.0007	0.0006	0.0005	0.0004	0.0004	-0.0001	-0.0001	0.0000	0.0000	0.0002	0.0003	0.0006	0.0008
16	0.0015	0.0013	0.0009	0.0009	0.0008	0.0006	0.0006	0.0005	0.0001	0.0000	-0.0001	-0.0001	0.0000	0.0002	0.0005	0.0007
5	0.0015	0.0013	0.0010	0.0009	0.0008	0.0006	0.0006	0.0006	0.0001	0.0000	-0.0001	-0.0001	0.0000	0.0002	0.0004	0.0007
11	0.0016	0.0014	0.0011	0.0010	0.0009	0.0007	0.0007	0.0007	0.0002	0.0002	0.0000	0.0000	-0.0001	0.0001	0.0003	0.0005
13	0.0018	0.0016	0.0013	0.0012	0.0011	0.0009	0.0009	0.0009	0.0004	0.0003	0.0002	0.0002	0.0001	-0.0001	0.0001	0.0004
7	0.0021	0.0019	0.0015	0.0015	0.0014	0.0012	0.0012	0.0011	0.0007	0.0006	0.0005	0.0004	0.0003	0.0001	-0.0001	0.0001
15	0.0023	0.0021	0.0018	0.0017	0.0016	0.0014	0.0014	0.0014	0.0009	0.0008	0.0007	0.0007	0.0005	0.0004	0.0001	-0.0001
23	0.0024	0.0022	0.0019	0.0018	0.0017	0.0015	0.0015	0.0015	0.0010	0.0010	0.0008	0.0008	0.0007	0.0005	0.0002	0.0000
26	0.0025	0.0023	0.0020	0.0019	0.0018	0.0016	0.0016	0.0016	0.0011	0.0010	0.0009	0.0009	0.0007	0.0005	0.0003	0.0001
17	0.0025	0.0024	0.0020	0.0019	0.0018	0.0017	0.0016	0.0016	0.0011	0.0011	0.0009	0.0009	0.0008	0.0006	0.0003	0.0001
32	0.0028	0.0026	0.0023	0.0022	0.0021	0.0019	0.0019	0.0019	0.0014	0.0013	0.0012	0.0012	0.0010	0.0008	0.0006	0.0004
18	0.0029	0.0027	0.0024	0.0023	0.0022	0.0020	0.0020	0.0020	0.0015	0.0014	0.0013	0.0013	0.0011	0.0009	0.0007	0.0005
20	0.0030	0.0028	0.0024	0.0024	0.0023	0.0021	0.0020	0.0020	0.0016	0.0015	0.0013	0.0013	0.0012	0.0010	0.0007	0.0005
28	0.0030	0.0028	0.0024	0.0024	0.0023	0.0021	0.0021	0.0020	0.0016	0.0015	0.0014	0.0013	0.0012	0.0010	0.0008	0.0005
25	0.0030	0.0028	0.0025	0.0024	0.0023	0.0021	0.0021	0.0021	0.0016	0.0016	0.0014	0.0014	0.0013	0.0011	0.0008	0.0006
19	0.0031	0.0029	0.0026	0.0025	0.0024	0.0022	0.0022	0.0022	0.0017	0.0016	0.0015	0.0015	0.0013	0.0012	0.0009	0.0007
27	0.0031	0.0030	0.0026	0.0025	0.0024	0.0023	0.0022	0.0022	0.0017	0.0017	0.0015	0.0015	0.0014	0.0012	0.0009	0.0007
22	0.0032	0.0030	0.0027	0.0026	0.0025	0.0023	0.0023	0.0023	0.0018	0.0017	0.0016	0.0016	0.0014	0.0013	0.0010	0.0008
30	0.0032	0.0030	0.0027	0.0026	0.0025	0.0023	0.0023	0.0023	0.0018	0.0018	0.0016	0.0016	0.0015	0.0013	0.0010	0.0008
24	0.0032	0.0031	0.0027	0.0027	0.0026	0.0024	0.0023	0.0023	0.0019	0.0018	0.0016	0.0016	0.0015	0.0013	0.0010	0.0008
21	0.0032	0.0031	0.0027	0.0027	0.0026	0.0024	0.0023	0.0023	0.0019	0.0018	0.0016	0.0016	0.0015	0.0013	0.0010	0.0008
29	0.0033	0.0031	0.0028	0.0027	0.0026	0.0024	0.0024	0.0024	0.0019	0.0018	0.0017	0.0017	0.0015	0.0013	0.0011	0.0009
31	0.0034	0.0032	0.0028	0.0028	0.0027	0.0025	0.0025	0.0024	0.0020	0.0019	0.0018	0.0017	0.0016	0.0014	0.0012	0.0009

Design Point	23	26	17	32	18	20	28	25	19	27	22	30	24	21	29	31
2	0.0024	0.0025	0.0025	0.0028	0.0029	0.0030	0.0030	0.0030	0.0031	0.0031	0.0032	0.0032	0.0032	0.0032	0.0033	0.0034
10	0.0022	0.0023	0.0024	0.0026	0.0027	0.0028	0.0028	0.0028	0.0029	0.0030	0.0030	0.0030	0.0031	0.0031	0.0031	0.0032
4	0.0019	0.0020	0.0020	0.0023	0.0024	0.0024	0.0024	0.0025	0.0026	0.0026	0.0027	0.0027	0.0027	0.0027	0.0028	0.0028
1	0.0018	0.0019	0.0019	0.0022	0.0023	0.0024	0.0024	0.0024	0.0025	0.0025	0.0026	0.0026	0.0027	0.0027	0.0027	0.0028
12	0.0017	0.0018	0.0018	0.0021	0.0022	0.0023	0.0023	0.0023	0.0024	0.0024	0.0025	0.0025	0.0026	0.0026	0.0026	0.0027
9	0.0015	0.0016	0.0017	0.0019	0.0020	0.0021	0.0021	0.0021	0.0022	0.0023	0.0023	0.0023	0.0024	0.0024	0.0024	0.0025
6	0.0015	0.0016	0.0016	0.0019	0.0020	0.0020	0.0021	0.0021	0.0022	0.0022	0.0023	0.0023	0.0023	0.0023	0.0024	0.0025
14	0.0015	0.0016	0.0016	0.0019	0.0020	0.0020	0.0020	0.0021	0.0022	0.0022	0.0023	0.0023	0.0023	0.0023	0.0024	0.0024
3	0.0010	0.0011	0.0011	0.0014	0.0015	0.0016	0.0016	0.0016	0.0017	0.0017	0.0018	0.0018	0.0019	0.0019	0.0019	0.0020
8	0.0010	0.0010	0.0011	0.0013	0.0014	0.0015	0.0015	0.0016	0.0016	0.0017	0.0017	0.0018	0.0018	0.0018	0.0018	0.0019
16	0.0008	0.0009	0.0009	0.0012	0.0013	0.0013	0.0014	0.0014	0.0015	0.0015	0.0016	0.0016	0.0016	0.0016	0.0017	0.0018
5	0.0008	0.0009	0.0009	0.0012	0.0013	0.0013	0.0013	0.0014	0.0015	0.0015	0.0016	0.0016	0.0016	0.0016	0.0017	0.0017
11	0.0007	0.0007	0.0008	0.0010	0.0011	0.0012	0.0012	0.0013	0.0013	0.0014	0.0014	0.0015	0.0015	0.0015	0.0015	0.0016
13	0.0005	0.0005	0.0006	0.0008	0.0009	0.0010	0.0010	0.0011	0.0012	0.0012	0.0013	0.0013	0.0013	0.0013	0.0013	0.0014
7	0.0002	0.0003	0.0003	0.0006	0.0007	0.0007	0.0008	0.0008	0.0009	0.0009	0.0010	0.0010	0.0010	0.0010	0.0011	0.0012
15	0.0000	0.0001	0.0001	0.0004	0.0005	0.0005	0.0005	0.0006	0.0007	0.0007	0.0008	0.0008	0.0008	0.0008	0.0009	0.0009
23	-0.0001	-0.0001	0.0000	0.0002	0.0003	0.0004	0.0004	0.0005	0.0006	0.0006	0.0006	0.0007	0.0007	0.0007	0.0007	0.0008
26	-0.0001	-0.0001	-0.0001	0.0002	0.0003	0.0003	0.0003	0.0004	0.0005	0.0005	0.0006	0.0006	0.0006	0.0006	0.0007	0.0007
17	0.0000	-0.0001	-0.0001	0.0001	0.0002	0.0003	0.0003	0.0004	0.0004	0.0005	0.0005	0.0006	0.0006	0.0006	0.0006	0.0007
32	0.0002	0.0002	0.0001	-0.0001	0.0000	0.0000	0.0001	0.0001	0.0002	0.0002	0.0003	0.0003	0.0003	0.0003	0.0004	0.0004
18	0.0003	0.0003	0.0002	0.0000	-0.0001	-0.0001	-0.0001	0.0000	0.0001	0.0001	0.0002	0.0002	0.0002	0.0002	0.0003	0.0003
20	0.0004	0.0003	0.0003	0.0000	-0.0001	-0.0001	-0.0001	-0.0001	0.0000	0.0000	0.0001	0.0001	0.0002	0.0002	0.0002	0.0003
28	0.0004	0.0003	0.0003	0.0001	-0.0001	-0.0001	-0.0001	-0.0001	0.0000	0.0000	0.0001	0.0001	0.0001	0.0001	0.0002	0.0003
25	0.0005	0.0004	0.0004	0.0001	0.0000	-0.0001	-0.0001	-0.0001	-0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002
19	0.0006	0.0005	0.0004	0.0002	0.0001	0.0000	0.0000	-0.0001	-0.0001	-0.0001	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001
27	0.0006	0.0005	0.0005	0.0002	0.0001	0.0000	0.0000	0.0000	-0.0001	-0.0001	-0.0001	-0.0001	0.0000	0.0000	0.0000	0.0001
22	0.0006	0.0006	0.0005	0.0003	0.0002	0.0001	0.0001	0.0001	0.0000	-0.0001	-0.0001	-0.0001	-0.0001	-0.0001	0.0000	0.0000
30	0.0007	0.0006	0.0006	0.0003	0.0002	0.0001	0.0001	0.0001	0.0000	-0.0001	-0.0001	-0.0001	-0.0001	-0.0001	-0.0001	0.0000
24	0.0007	0.0006	0.0006	0.0003	0.0002	0.0002	0.0001	0.0001	0.0000	0.0000	-0.0001	-0.0001	-0.0001	-0.0001	-0.0001	0.0000
21	0.0007	0.0006	0.0006	0.0003	0.0002	0.0002	0.0001	0.0001	0.0000	0.0000	-0.0001	-0.0001	-0.0001	-0.0001	-0.0001	0.0000
29	0.0007	0.0007	0.0006	0.0004	0.0003	0.0002	0.0002	0.0001	0.0001	0.0000	0.0000	-0.0001	-0.0001	-0.0001	-0.0001	-0.0001
31	0.0008	0.0007	0.0007	0.0004	0.0003	0.0003	0.0003	0.0002	0.0001	0.0001	0.0000	0.0000	0.0000	0.0000	-0.0001	-0.0001

Appendix B.2 Tukey- Kramer Multiple Comparison.

Positive values show pairs of means that are significantly different.

Table 1 Tukey-Kramer Multiple Comparison

Design Point	2	10	4	1	12	9	6	14	3	8	16	5	11	13	7	15
2	-0.0003	-0.0001	0.0003	0.0003	0.0004	0.0006	0.0007	0.0007	0.0011	0.0012	0.0013	0.0014	0.0015	0.0017	0.0019	0.0022
10	-0.0001	-0.0003	0.0001	0.0002	0.0003	0.0004	0.0005	0.0005	0.0010	0.0010	0.0012	0.0012	0.0013	0.0015	0.0018	0.0020
4	0.0003	0.0001	-0.0003	-0.0002	-0.0001	0.0001	0.0001	0.0001	0.0006	0.0007	0.0008	0.0008	0.0010	0.0012	0.0014	0.0016
1	0.0003	0.0002	-0.0002	-0.0003	-0.0002	0.0000	0.0001	0.0001	0.0005	0.0006	0.0008	0.0008	0.0009	0.0011	0.0014	0.0016
12	0.0004	0.0003	-0.0001	-0.0002	-0.0003	-0.0001	0.0000	0.0000	0.0004	0.0005	0.0007	0.0007	0.0008	0.0010	0.0013	0.0015
9	0.0006	0.0004	0.0001	0.0000	-0.0001	-0.0003	-0.0002	-0.0002	0.0003	0.0003	0.0005	0.0005	0.0006	0.0008	0.0011	0.0013
6	0.0007	0.0005	0.0001	0.0001	0.0000	-0.0002	-0.0003	-0.0003	0.0002	0.0003	0.0004	0.0005	0.0006	0.0008	0.0010	0.0013
14	0.0007	0.0005	0.0001	0.0001	0.0000	-0.0002	-0.0003	-0.0003	0.0002	0.0003	0.0004	0.0004	0.0006	0.0008	0.0010	0.0012
3	0.0011	0.0010	0.0006	0.0005	0.0004	0.0003	0.0002	0.0002	-0.0003	-0.0002	0.0000	0.0000	0.0001	0.0003	0.0006	0.0008
8	0.0012	0.0010	0.0007	0.0006	0.0005	0.0003	0.0003	0.0003	-0.0002	-0.0003	-0.0001	-0.0001	0.0000	0.0002	0.0005	0.0007
16	0.0013	0.0012	0.0008	0.0008	0.0007	0.0005	0.0004	0.0004	0.0000	-0.0001	-0.0003	-0.0002	-0.0001	0.0001	0.0003	0.0006
5	0.0014	0.0012	0.0008	0.0008	0.0007	0.0005	0.0005	0.0004	0.0000	-0.0001	-0.0002	-0.0003	-0.0001	0.0001	0.0003	0.0005
11	0.0015	0.0013	0.0010	0.0009	0.0008	0.0006	0.0006	0.0006	0.0001	0.0000	-0.0001	-0.0001	-0.0003	-0.0001	0.0002	0.0004
13	0.0017	0.0015	0.0012	0.0011	0.0010	0.0008	0.0008	0.0008	0.0003	0.0002	0.0001	0.0001	-0.0001	-0.0003	0.0000	0.0002
7	0.0019	0.0018	0.0014	0.0014	0.0013	0.0011	0.0010	0.0010	0.0006	0.0005	0.0003	0.0003	0.0002	0.0000	-0.0003	0.0000
15	0.0022	0.0020	0.0016	0.0016	0.0015	0.0013	0.0013	0.0012	0.0008	0.0007	0.0006	0.0005	0.0004	0.0002	0.0000	-0.0003
23	0.0023	0.0021	0.0018	0.0017	0.0016	0.0014	0.0014	0.0014	0.0009	0.0008	0.0007	0.0007	0.0005	0.0003	0.0001	-0.0001
26	0.0024	0.0022	0.0018	0.0018	0.0017	0.0015	0.0014	0.0014	0.0010	0.0009	0.0008	0.0007	0.0006	0.0004	0.0002	-0.0001
17	0.0024	0.0022	0.0019	0.0018	0.0017	0.0015	0.0015	0.0015	0.0010	0.0009	0.0008	0.0008	0.0006	0.0005	0.0002	0.0000
32	0.0027	0.0025	0.0021	0.0021	0.0020	0.0018	0.0017	0.0017	0.0013	0.0012	0.0011	0.0010	0.0009	0.0007	0.0005	0.0002
18	0.0028	0.0026	0.0022	0.0022	0.0021	0.0019	0.0018	0.0018	0.0014	0.0013	0.0012	0.0011	0.0010	0.0008	0.0006	0.0003
20	0.0028	0.0027	0.0023	0.0022	0.0021	0.0020	0.0019	0.0019	0.0014	0.0014	0.0012	0.0012	0.0011	0.0009	0.0006	0.0004
28	0.0029	0.0027	0.0023	0.0023	0.0022	0.0020	0.0019	0.0019	0.0015	0.0014	0.0012	0.0012	0.0011	0.0009	0.0006	0.0004
25	0.0029	0.0027	0.0024	0.0023	0.0022	0.0020	0.0020	0.0020	0.0015	0.0014	0.0013	0.0013	0.0011	0.0009	0.0007	0.0005
19	0.0030	0.0028	0.0024	0.0024	0.0023	0.0021	0.0021	0.0020	0.0016	0.0015	0.0014	0.0013	0.0012	0.0010	0.0008	0.0005
27	0.0030	0.0028	0.0025	0.0024	0.0023	0.0021	0.0021	0.0021	0.0016	0.0015	0.0014	0.0014	0.0013	0.0011	0.0008	0.0006
22	0.0031	0.0029	0.0025	0.0025	0.0024	0.0022	0.0022	0.0021	0.0017	0.0016	0.0015	0.0014	0.0013	0.0011	0.0009	0.0006
30	0.0031	0.0029	0.0026	0.0025	0.0024	0.0022	0.0022	0.0022	0.0017	0.0016	0.0015	0.0015	0.0013	0.0011	0.0009	0.0007
24	0.0031	0.0029	0.0026	0.0025	0.0024	0.0022	0.0022	0.0022	0.0017	0.0017	0.0015	0.0015	0.0014	0.0012	0.0009	0.0007
21	0.0031	0.0029	0.0026	0.0025	0.0024	0.0022	0.0022	0.0022	0.0017	0.0017	0.0015	0.0015	0.0014	0.0012	0.0009	0.0007
29	0.0032	0.0030	0.0026	0.0026	0.0025	0.0023	0.0022	0.0022	0.0018	0.0017	0.0016	0.0015	0.0014	0.0012	0.0010	0.0007
31	0.0032	0.0031	0.0027	0.0027	0.0026	0.0024	0.0023	0.0023	0.0019	0.0018	0.0016	0.0016	0.0015	0.0013	0.0010	0.0008

Design Point	23	26	17	32	18	20	28	25	19	27	22	30	24	21	29	31
2	0.0023	0.0024	0.0024	0.0027	0.0028	0.0028	0.0029	0.0029	0.0030	0.0030	0.0031	0.0031	0.0031	0.0031	0.0032	0.0032
10	0.0021	0.0022	0.0022	0.0025	0.0026	0.0027	0.0027	0.0027	0.0028	0.0028	0.0029	0.0029	0.0029	0.0029	0.0030	0.0031
4	0.0018	0.0018	0.0019	0.0021	0.0022	0.0023	0.0023	0.0024	0.0024	0.0025	0.0025	0.0026	0.0026	0.0026	0.0026	0.0027
1	0.0017	0.0018	0.0018	0.0021	0.0022	0.0022	0.0023	0.0023	0.0024	0.0024	0.0025	0.0025	0.0025	0.0025	0.0026	0.0027
12	0.0016	0.0017	0.0017	0.0020	0.0021	0.0021	0.0022	0.0022	0.0023	0.0023	0.0024	0.0024	0.0024	0.0024	0.0025	0.0026
9	0.0014	0.0015	0.0015	0.0018	0.0019	0.0020	0.0020	0.0020	0.0021	0.0021	0.0022	0.0022	0.0022	0.0022	0.0023	0.0024
6	0.0014	0.0014	0.0015	0.0017	0.0018	0.0019	0.0019	0.0020	0.0021	0.0021	0.0022	0.0022	0.0022	0.0022	0.0022	0.0023
14	0.0014	0.0014	0.0015	0.0017	0.0018	0.0019	0.0019	0.0020	0.0020	0.0021	0.0021	0.0022	0.0022	0.0022	0.0022	0.0023
3	0.0009	0.0010	0.0010	0.0013	0.0014	0.0014	0.0015	0.0015	0.0016	0.0016	0.0017	0.0017	0.0017	0.0017	0.0018	0.0019
8	0.0008	0.0009	0.0009	0.0012	0.0013	0.0014	0.0014	0.0014	0.0015	0.0015	0.0016	0.0016	0.0017	0.0017	0.0017	0.0018
16	0.0007	0.0008	0.0008	0.0011	0.0012	0.0012	0.0012	0.0013	0.0014	0.0014	0.0015	0.0015	0.0015	0.0015	0.0016	0.0016
5	0.0007	0.0007	0.0008	0.0010	0.0011	0.0012	0.0012	0.0013	0.0013	0.0014	0.0014	0.0015	0.0015	0.0015	0.0015	0.0016
11	0.0005	0.0006	0.0006	0.0009	0.0010	0.0011	0.0011	0.0011	0.0012	0.0013	0.0013	0.0013	0.0014	0.0014	0.0014	0.0015
13	0.0003	0.0004	0.0005	0.0007	0.0008	0.0009	0.0009	0.0009	0.0010	0.0011	0.0011	0.0011	0.0012	0.0012	0.0012	0.0013
7	0.0001	0.0002	0.0002	0.0005	0.0006	0.0006	0.0006	0.0007	0.0008	0.0008	0.0009	0.0009	0.0009	0.0009	0.0010	0.0010
15	-0.0001	-0.0001	0.0000	0.0002	0.0003	0.0004	0.0004	0.0005	0.0005	0.0006	0.0006	0.0007	0.0007	0.0007	0.0007	0.0008
23	-0.0003	-0.0002	-0.0002	0.0001	0.0002	0.0003	0.0003	0.0003	0.0004	0.0005	0.0005	0.0005	0.0006	0.0006	0.0006	0.0007
26	-0.0002	-0.0003	-0.0002	0.0000	0.0001	0.0002	0.0002	0.0003	0.0004	0.0004	0.0004	0.0005	0.0005	0.0005	0.0005	0.0006
17	-0.0002	-0.0002	-0.0003	0.0000	0.0001	0.0002	0.0002	0.0002	0.0003	0.0003	0.0004	0.0004	0.0005	0.0005	0.0005	0.0006
32	0.0001	0.0000	0.0000	-0.0003	-0.0002	-0.0001	-0.0001	0.0000	0.0001	0.0001	0.0002	0.0002	0.0002	0.0002	0.0002	0.0003
18	0.0002	0.0001	0.0001	-0.0002	-0.0003	-0.0002	-0.0002	-0.0001	-0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002
20	0.0003	0.0002	0.0002	-0.0001	-0.0002	-0.0003	-0.0002	-0.0002	-0.0001	-0.0001	0.0000	0.0000	0.0000	0.0000	0.0001	0.0002
28	0.0003	0.0002	0.0002	-0.0001	-0.0002	-0.0002	-0.0003	-0.0002	-0.0001	-0.0001	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001
25	0.0003	0.0003	0.0002	0.0000	-0.0001	-0.0002	-0.0002	-0.0003	-0.0002	-0.0002	-0.0001	-0.0001	0.0000	0.0000	0.0000	0.0001
19	0.0004	0.0004	0.0003	0.0001	-0.0001	-0.0001	-0.0001	-0.0002	-0.0003	-0.0002	-0.0002	-0.0002	-0.0001	-0.0001	-0.0001	0.0000
27	0.0005	0.0004	0.0003	0.0001	0.0000	-0.0001	-0.0001	-0.0002	-0.0002	-0.0003	-0.0002	-0.0002	-0.0002	-0.0002	-0.0001	0.0000
22	0.0005	0.0004	0.0004	0.0002	0.0001	0.0000	0.0000	-0.0001	-0.0002	-0.0002	-0.0003	-0.0002	-0.0002	-0.0002	-0.0002	-0.0001
30	0.0005	0.0005	0.0004	0.0002	0.0001	0.0000	0.0000	-0.0001	-0.0002	-0.0002	-0.0002	-0.0003	-0.0002	-0.0002	-0.0002	-0.0001
24	0.0006	0.0005	0.0005	0.0002	0.0001	0.0000	0.0000	0.0000	-0.0001	-0.0002	-0.0002	-0.0002	-0.0003	-0.0003	-0.0002	-0.0001
21	0.0006	0.0005	0.0005	0.0002	0.0001	0.0000	0.0000	0.0000	-0.0001	-0.0002	-0.0002	-0.0002	-0.0003	-0.0003	-0.0002	-0.0001
29	0.0006	0.0005	0.0005	0.0002	0.0001	0.0001	0.0001	0.0000	-0.0001	-0.0001	-0.0002	-0.0002	-0.0002	-0.0002	-0.0003	-0.0002
31	0.0007	0.0006	0.0006	0.0003	0.0002	0.0002	0.0001	0.0001	0.0000	0.0000	-0.0001	-0.0001	-0.0001	-0.0001	-0.0002	-0.0003

Appendix B.3 Ranking and Selection

Table 6 Ranking and Selection

Design Setting	Crossover	Selection	Population Size	Probability of Crossover	Probability of Mutation	Number Replaced	P
31	uniform	roulette	100	0.95	0.03	2	0.1682497
29	uniform	tournament	100	0.95	0.03	25	0.124204
21	uniform	tournament	100	0.85	0.03	2	0.0906266
24	two-point	roulette	100	0.85	0.03	2	0.084916
30	two-point	tournament	100	0.95	0.03	2	0.0764209
22	two-point	tournament	100	0.85	0.03	25	0.0735273
27	uniform	roulette	50	0.95	0.03	25	0.0554826
19	uniform	roulette	50	0.85	0.03	2	0.0449968
25	uniform	tournament	50	0.95	0.03	2	0.0335378
28	two-point	roulette	50	0.95	0.03	2	0.0318686
23	uniform	roulette	100	0.85	0.03	25	0.0299966
20	two-point	roulette	50	0.85	0.03	25	0.0291113
18	two-point	tournament	50	0.85	0.03	2	0.0284038
32	two-point	roulette	100	0.95	0.03	25	0.0221866
15	uniform	roulette	100	0.95	0.01	25	0.0109189
17	uniform	tournament	50	0.85	0.03	25	0.0102421
26	two-point	tournament	50	0.95	0.03	25	0.0101765
7	uniform	roulette	100	0.85	0.01	2	0.0100477
13	uniform	tournament	100	0.95	0.01	2	0.0084658
16	two-point	roulette	100	0.95	0.01	2	0.007204
11	uniform	roulette	50	0.95	0.01	2	0.0069814
8	two-point	roulette	100	0.85	0.01	25	0.0059804
5	uniform	tournament	100	0.85	0.01	25	0.005803
3	uniform	roulette	50	0.85	0.01	25	0.004433
6	two-point	tournament	100	0.85	0.01	2	0.0042146
1	uniform	tournament	50	0.85	0.01	2	0.0037717
4	two-point	roulette	50	0.85	0.01	2	0.0036967
9	uniform	tournament	50	0.95	0.01	25	0.0035382
14	two-point	tournament	100	0.95	0.01	25	0.0034003
10	two-point	tournament	50	0.95	0.01	2	0.0028188
12	two-point	roulette	50	0.95	0.01	25	0.0026511
2	two-point	tournament	50	0.85	0.01	25	0.0021274

Probability of Correctly Selection is greater than 0.95

APPENDIX C Genetic Algorithm In C++

```
/* ------------------------------------------------------------------
--------
 RANDOM FEASIBLE  GENETIC ALGORITHM FOR MKP.
------------------------------------------------------------------
----- */
#include <stdio.h>
#include <iostream.h>
#include <ctime>
#include "GASState.h"
#include "GA1DBinS.h"
float Objective(GAGenome &);  // This is the declaration of our obj
function.
                        // The definition comes later in the file.
#define NO_VAR 101
#define MKP_FILE "gr1120.txt"// Problem file

float c[NO_VAR],a1[NO_VAR],a2[NO_VAR];
float b1,b2,ZIP,Rho,Slackness;
int  dropper,adder;
// seed is optional but better use to have same results each time you
run the code
// Seeding is important in random search
//int seed = 12321;
int
main()
{
      // opening data file
      ifstream in(MKP_FILE);
  char zipi[32] = "ALO.txt";
      ofstream outfile ;
      outfile.open(zipi, (ios::out | ios::app ));

  for (int jy=1; jy<3  ; jy++)


  {
        dropper=adder = 0;

            double dump;
          int prob;


     if(!in) {
           cerr << "could not read data file " << MKP_FILE << "\n";
                 exit(1);
             }
           // first rows of each problem is read to have parameters
and RHS of problem
           //the parameters of the program.
             in >> prob;
             in >> dump;
```

```
            in >> dump;
            in >> ZIP;
            in >> b1;
            in >> b2;
            in >> Rho;
            in >> Slackness;
           in >> dump;
            in >> dump;
            in >> dump;

            cout << jy <<  "\n";
            int nvar = 0;
            do{                    // reading obj. func. values

                  in >> c[nvar];
                 nvar++;
            }
               while(nvar<100);
                 nvar = 0;
           do{                             //reading first constraint

                   in >> a1[nvar];
                   nvar++;
            }
               while(nvar<100);

           nvar = 0;
                 do{                       //reading second constraint

               in >> a2[nvar];
                nvar++;
            }
               while(nvar<100);

                 if (in.eof())
                       in.close();


            if(nvar >= NO_VAR) {
                cerr << "data file contains more VARIABLES than
allowed for in the fixed\n";
                cerr << "arrays.  Recompile the program with larger
arrays or try a\n";
                cerr << "smaller problem.\n";
                exit(1);
            }
          // Declare variables for the GA parameters and set them to
some default values
            int length   = 100;
           int  flush = 5;
           int  atama = 10000;

           int  popsize  = 50;
            int ngen     = 10000;
            float pmut   = 0.03;
```

```
              double pcross = 0.95;
             ;
                  int best = 5;
                  time_t      time1;
                  time1=0;
                  time1 = clock();


              GA1DBinaryStringGenome genome(length, Objective);
              GASteadyStateGA ga(genome);
              ga.initialize();

            ga.populationSize(popsize);
             ga.nGenerations(ngen);
             ga.pMutation(pmut);
              ga.nReplacement(2);
              ga.pCrossover(pcross);
              ga.flushFrequency(flush);
              ga.scoreFrequency(atama);
              ga.scoreFilename("Graphicin.txt");

      ga.crossover(GA1DBinaryStringGenome :: TwoPointCrossover);
                  //ga.crossover(GA1DBinaryStringGenome ::
OnePointCrossover);
                  //ga.crossover(GA1DBinaryStringGenome ::
EvenOddCrossover );*/
                  //ga.crossover(GA1DBinaryStringGenome ::
StringtoChangeCrossover );
            //    ga.crossover(GA1DBinaryStringGenome ::
UniformCrossover );

      //ga.terminator(GAGeneticAlgorithm::TerminateUponZipConvergence);
                  //  ga.parameters("settings.txt");
            // ga.parameters(argc,argv);
            GARouletteWheelSelector tournement; //RankSelector  ;
RoulletteWheelSelector ;
            ga.selector(tournement)        ;
            GASigmaTruncationScaling sigma          ;
            ga.scaling(sigma)              ;
               ga.selectScores(2);
            // ga.statistics().write(outfile);
            //genome=ga.generation();
             ga.evolve();
       // genome = ;



              if (ga.done){
                    time_t time2;
                    time2=0;
                    time2 = clock();
                    cout << dropper << "\t" <<adder<< "\t" <<
ga.statistics().crossovers()<< "\n";
                    outfile << difftime(time2,time1)/ CLOCKS_PER_SEC <<
"\t";
```

```
            //outfile << ga.statistics().online();
                        //cout << genome;
                    }
            // GAParameterlist params ;
            // params.set(gaNscoreFilename, "out.dat") ;



}
      return 0;    }


// Objective function. The most imoratant part. Basically a child is
evaluated by this function before
//getting into the population

             float
           Objective(GAGenome& g) {
             GA1DBinaryStringGenome & genome = (GA1DBinaryStringGenome
&)g;

 //*char Zip[32] = "ONLINE.txt";
//    ofstream outfile ;
//    outfile.open(Zip, (ios::out | ios::app));
//    outfile << genome <<"\n";

             float score=0.0;
             float feasible1=0.0;// left hand side of current first
constraint
             float feasible2=0.0;// left hand side of  current second
constraint

             //* First constraints are calculated to see if the
reported solution is good!
             for(int i=0; i<genome.length(); i++)
      {
                                  feasible1 += genome.gene(i)*a1[i];
                                    feasible2 += genome.gene(i)*a2[i];

                        }


             //* if the current solution is not good ( violated
constraint(s) ), then DROP and ADD algorithms are
            // invoked to make child good for the feasible population.
This is done by starting from a random point
            // in its chromosome '1' are excluded until they are
feasible in terms of both constraints.

             if ((feasible1>b1) || (feasible2>b2)){

                   dropper++ ;
```

```
                              int mdpnt =
GARandomInt(0,(genome.length()-1));
                              for (int q=mdpnt ; q<genome.length() ;
q++){
                         if(genome.gene(q)){

      genome.gene(q,0);
                                                  feasible1 -= a1[q];
                                                  feasible2 -= a2[q];

      }
                         if ((feasible1<=b1) && (feasible2<=b2)) break;

                         if ( q == (genome.length()-1)){

                                    for (int j=0 ; j<mdpnt ; j++){

      if(genome.gene(j)){

      genome.gene(j,0);

      feasible1 -= a1[j];

      feasible2 -= a2[j];

                         }

                         if ((feasible1<=b1) && (feasible2<=b2)) break;}

                         }


                         }


       }
  //* After DROP algortihm, ADD algorithm tries to improve the quality
of  chromosome by
 // adding feasible items ,starting from a random byte.


  int addition = GARandomInt(0,(genome.length()-1));
for (int t=addition; t<genome.length() ; t++){
            if(((feasible1 +a1[t])<=b1) && ((feasible2 + a2[t])<=b2) &&
(!(genome.gene(t)))){
                  adder++;
                                          genome.gene(t,1);
                                          feasible1 += a1[t];
                                          feasible2 += a2[t];}

      if ((feasible1==b1) && (feasible2==b2)) break;

      if ( t == (genome.length()-1)){
```

```
      for (int j=0 ; j<addition ; j++){
            if(((feasible1 +a1[j])<=b1) && ((feasible2 + a2[j])<=b2) &&
      (!(genome.gene(j)))){
                                                adder++;
                                          genome.gene(j,1);
                                          feasible1 += a1[j];
                                          feasible2 += a2[j];}

                  if ((feasible1==b1) && (feasible2==b2)) break;}
      }

                                          }



for(int w=0; w<genome.length(); w++){ // Score of the child is returned
      score +=  genome.gene(w)* c[w];}


        return score;


              }
```

Bibliography

Balas, Egon. "An Additive Algorithm for Solving Linear Programs with Zero-one Variables," Operations Research, 13: 517-546 (1965).

Balas, E and Martin, C.H. "Pivot and Complement – A heuristic for 0-1 Programming," Management Science, 26(1): 86-96 (1980).

Barr, R.S., J.P. Kelly, B.L. Golden, M.G.C. Resende and W.R. Steward Jr. "Designing and Reporting on Computational Experiments with Heuristic Methods," Journal of Heuristics, 1(1): 9-32 (1995).

Beasley, J.E., D.R. Bull, R.R. Martin. "An Overview of Genetic Algorithms: fundamentals and research topics," University Computing, 15: 58-69 and 170-181 (1993).

Bjorndal, M.H, A. Caprara, P.I. Cowling, Della Croce, H. Lourencho, F. Malucelli, A.J. Orman, D. Pisinger, C. Rego, J.J. Salazar. " Some Thoughts on Combinatorial Optimization," European Journal of Operational Sciences, 83: 253-270 (1995).

Chu, P.C and J.E. Beasley. "A Genetic Algorithm for the Multiconsraint Knapsack Problem," The Management School, Imperial College, Working Paper http:\mscmga.ms.ic.ac.uk/pchu/pchu.html, September 1998.

Dammayer, F. and S. Voss. "Dynamic tabu list management using reverse elimination method,' Annals of Operations Research, 41: 31-46 (1993).

Davis, Lawrence. Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold, 1991.

Drexl, A. "A Simulated Annealing Approach to the Multiconstraint Zero-One Knapsack problem," Computing,40: 1-8 (1988).

Fisher, M., H.G. Alexander and Kan Rinnkoy. "The Design, Analysis and Implementation of Heuristics," Management Sciences, 34(3): 263-265 (1988).

Frieze, A. M. and M. R. B. Clarke. "Approximation algorithms for the m-dimensional 0-1 Knapsack Problem: Worst-case and probabilistic analyses," European Journal of Operations Research, 15: 100-109 (1984).

Goldberg, David. Genetic Algorithms in Search, optimization and machine Learning. Mass.:Addison-Wesley,1989.

Hanafi, S, A. Freville and A. El Abedellaoui."Comparison of Heuristics for the 0-1 Multidimensional Knapsack Problem," In : Meta-heuristics: theory and Applications. Boston: Kluwer Academics, 1996.

Hanafi ,S. and A. Freville. "An Efficient Tabu Search for the Multiconstraint Zero-One Knapsack Problem," European Journal of Operational Research, 106: 659-675 (1998).

Hill, R.R. and C.H Reilly. "The Effects of Coefficient Correlation Structure in Two-Dimensional Knapsack Problems on Solution Procedure Performance," Working Paper, Air Force Institute of Technology, W.P.AFB, Dayton, Ohio, U.S.A., 1997.

Hill, R. "An Analytical Comparison of Optimization Problem Generation Methodologies." Proceeding of the 1998 Winter Simulation Conference. 609-615. Washington DC.: Institude of Electrical and Electronics Engineers, 1998

Hoff, Arrild, Arne Lokketangen and Ingvar Mittet. "Genetic Algorithms for 0/1 Multidimensional Knapsack Problems," Working Paper, Molde College, Britveien 2, 6400 Molde, Norway, 1998.

Hoffman, Karla L. and Manfred Padberg."Combinatorial and Integer Programming," In: Encyclopedia of Operations Research and Management Science. Gauss, Saul and Carl L. Harris. Norwell: Kluwer Academic Publishers, 1996.

Hooker, J.N. "Needed: An Empirical Science of Algorithms," Operations Research, 42(2): 201-212 (1994).

Kochenberger,Gary, B.A. McCarl and F.P. Wymann. "A Heuristic for General Integer Programming," Decision Science,5: 36-44 (1974).

Lee, J.S and M. Guignard. "An Approximate Algorithm for Multidimensional Zero-One Knapsack Problems- a Parametric Approach," Management Science,34: 402-410 (1988).

Lokketangen, Arne. "A Comparison of a Genetic and Tabu Search Method for 0/1 Multidimensional Knapsack Problems". Working Paper, Molde College, Britveien 2, 6400 Molde, Norway. 1997.

Lokketangen, A. and F. Glover."Solving Zero-One Mixed Integer Programming Problems Using Tabu Search," European Journal of Heuristics, 106: 624-658 (1998).

Loulou, R. and E. Michalides. "New Greedy Heuristics for 0-1 Multidimensional 0-1 Knapsack Problem," Operations Research, 27 (6): 1101-1114 (1979).

Magazine, M.J. and O. Oguz."A heuristics Algorithm for the Multidimensional Zero-One Knapsack Problem," European Journal of Heuristics,16: 319-326 (1984).

Martello, S. and P. Toth. Knapsack Problems : algorithms and computer implementations. New York: J.Wiley & Sons, 1990.

Michalewicz, Zbigniew. Genetic Algorithms + Data Structures = Evolution Programs.New York: Springer-Verlag, 1992.

Nemhauser, G. and Laurence W.A. Integer and Combinatorial Optimization. John Wiley& Sons Inc. 1988

Osman, I. and J. Kelly. Meta-Heuristics : theory & applications. Boston: Kluwer Academics, 1996.

Parker, gary R. and Ronald L. Rardin. Discrete Optimization. Boston: Academic Press Inc., 1988.

Reeves, Colin R. Modern Heuristic Techniques for Combinatorial Problems. New York: John Wiley & Sons, 1993

Reingold, Edward,M. and others, Combinatorial Algorithms: Theory and Applications, New Jersey : Prentince-Hall Inc. 1977

Petersen, Clifford. "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R&D Projects," Management Science, 13: 736-745 (1967).

Schaffer, J.David, Richard A. Caruana, Larry J. Eshelman and Rajarshi Das."A Study of Control Parameters Affecting Online performance of Genetic Algorithms for Function Optimization," In: Proceedings of the Third International Conference on Genetic Algorithms. Schaffer, D.A and others. San Mateo: George Mason University, 1994.

Senju, S. and Y. Toyoda. "An approach to linear programming with 0-1 variables," Management Sciences, 15: 196-207 (1968).

Shih, W. "A branch and bound method for multiconstraint zero-one knapsack problem," Journal of the Operational Research Society, 30: 369-378 (1979).

Silver, Edward, R.V.V. Vidal and Dominique de Werra. "A tutorial on Heuristic methods," European Journal of Operational Research, 5: 153-162 (1980).

Syswerda, G. "Uniform Crossover in genetic Algorithms," In: Proceedings of the Third International Conference on Genetic Algorithms. Schaffer, D.A and others. San Mateo: George Mason University, 1994.

Theil, J. and Voss, S. "Some Experiences on Solving Multiconstraint 0-1 Knapsack problems with Genetic Algorithms," Computing,32: 226-242 (December 1993).

Toyoda, Y. "A simplified algorithm for obtaining approximate solutions to zero-one programming problems," Management Sciences, 21: 1417-1427 (1975).

Whitley, Darrell. "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best," In: Proceedings of the Third International Conference on Genetic Algorithms. Schaffer, D.A and others. San Mateo: George Mason University, 1994.

Zanakis, S. and J. Evans. "Heuristic optimization: Why, When and How to Use It," Interfaces,11: 75-82 (October 1981).

Zanakis, Stelios. "Heuristic 0-1 Linear Programming: An experimental Comparison of Three Methods," Management Science, 24: 91-104 (1977).

VITA

Lieutenant Mehmet Eravsar was born on 28 April 1971 in Kayseri, Turkey. He graduated from Kuleli Military High School in 1989, from Turkish Air Force Academy in 1993, in Istanbul. He successfully finished his Undergraduate Pilot Training in January 1995 and assigned to 133rd.Combat Readiness Flight Squadron where he was recognized as Distinguished Graduate. Upon graduation, he completed F-16 Training Squadron and became combat ready pilot in 153rd. Fighter Squadron in Merzifon. Upon graduation his next assignment is not known.

Acknowledgements

First of all, I would like to thank the Turkish People whose taxes paid for my top of the line OR education and let me meet high caliber officers and friends at AFIT.

A Turkish proverb says, "Volunteer to be a slave for forty years for someone who teaches even a single letter." Without exaggeration, this proverb is not even close to describe my feelings and wishes for the faculty who worked hard to teach and stood as a vivid example of professionalism. Among the faculty special thanks to my advisors, Maj. Hill and Dr. Moore, who not only helped me with their expertise in OR but with their never ending thoughtfulness and friendliness. Maj Hill, despite his "already full white-board," was very generous and thoughtful to offer a worthwhile thesis topic. His assistance was vital to my graduating.

Finally, among all the wonderful people I met, I want to thank the one who really sacrificed the most, my wife Esra. We got married the day I left Turkey to come to AFIT. Since then she allowed me to spend the most valuable days of our lives at AFIT while she was alone and separated from me. There is nothing I can do to make up for the lost time, but I promise I will try.

My next project is to construct a real-world model which will remember all these wonderful people and their efforts to work for peace, humanity and friendship. This will keep me busy to the last day of my life.

Table of Contents

List of Figures

List of Tables

Abstract

Meta-heuristics have been deployed to solve many hard combinatorial and optimization problems. Parameterization of meta-heuristics is an important challenging aspect of meta-heuristic use since many of the features of these algorithms can not be explained theoretically. Experiences with Genetic Algorithms (GA) applied to Multidimensional Knapsack Problems (MKP) have shown that this class of algorithm is very sensitive to parameterization. Many studies use standard test problems, which provide a firm basis for study comparisons but ignore the effect of problem correlation structure.

This thesis applies GA to MKP. A new random repair operator, which projects infeasible solutions into feasiblity, is proposed. This GA application is tested with synthetic test problems, which map possible correlation structures as well as possible slackness settings. Effect of correlation structure on solution quality found both statistically and practically significant. Depending on the Response Surface Methodology design, proposed is a GA parameter setting which is robust in both solution quality and computation time.

A COMPARISON OF GENETIC ALGORITHM PARAMETERIZATION ON SYNTHETIC TEST PROBLEMS

Chapter 1. Motivation and Outlines

Meta-heuristics have been used to solve many hard combinatorial and optimization problems. Parameterizations of meta-heuristics are an important and challenging aspect of meta-heuristic use since many of the features of these algorithms can not be explained theoretically. Deficiencies in analytic approaches mean empirical studies are conducted to examine parameterization of meta-heuristics. This emerging "empirical science" is addressed in Hooker (1994).

Experience with Genetic Algorithms (GA) has shown that this class of algorithm is very sensitive to parameterization. This has been true in GA studies applied to Multidimensiona

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1999	Master's Thesis

4. TITLE AND SUBTITLE
A COMPARISON OF GENETIC ALGORITHMS' PARAMETERIZATION ON SYNTHETIC OPTIMIZATION PROBLEMS

5. FUNDING NUMBERS

6. AUTHOR(S)

Mehmet Eravsar, Lieutenant, TUAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
2950 P Street
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/99M-05

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
2950 P Street
WPAFB OH 45433-7765

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Raymond R. Hill, Major, USAF

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Meta-heuristics have been deployed to solve many hard combinatorial and optimization problems. Parameterization of meta-heuristics is an important challenging aspect of meta-heuristic use since many of the features of these algorithms can not be explained theoretically. Experiences with Genetic Algorithms (GA) applied to Multidimensional Knapsack Problems (MKP) have shown that this class of algorithm is very sensitive to parameterization. Many studies use standard test problems, which provide a firm basis for study comparisons but ignore the effect of problem correlation structure.

This thesis applies GA to MKP. A new random repair operator, which projects infeasible solutions into feasible region, is proposed. This GA application is tested with synthetic test problems, which map possible correlation structures as well as possible slackness settings. Effect of correlation structure on solution quality found both statistically and practically significant. Depending on the Response Surface Methodology design, proposed is a GA parameter setting which is robust in both solution quality and computation time.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Heuristics, Meta-heuristics, Genetic Algorithms, Knapsack Problems, Multidimensional Knapsack Problems	78
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL